1

第1章　对象和误差

1.1

1.1　计算方法的对象与特点

1.1.1　研究的对象

计算方法是数学的一个分支，研究的对象是求解各种数学问题的数值方法及有关理论．内容包括函数的数值逼近（代数插值与最佳逼近），数值积分与数值微分，非线性(代数的与超越的）方程的数值解法，数值线代数（线性代数方程组的解法与矩阵特征值问题的计算)，常微分方程的数值解法及偏微分方程的数值解法等．

计算方法也称数值分析，它所以成为数学中的独立分支，一方面是数学本身的发展为之提供了可能，另一方面是40年代电子计算机的问世使之成为必要．在数学发展中，理论和计算是紧密联系的，逐渐积累了越来越多的数值方法．因此，计算方法中不少方法是传统的（一些方法的命名就表明了这一点），它们在19世纪或18世纪，甚至更早一些时候就已建立．现代计算机的出现为大规模的数值计算创造了条件，集中而系统地研究适用于计算机的数值方法立即变得十分迫切和必要．计算方法并不仅仅是一些数值方法的简单积累，而且揭示包含在多种多样的数值方法之间的相同的结构和统一的原理．它在大量的数值计算实践和理论分析工作的基础上迅速发展着，原有的方法有的使用至今，有的逐步被淘汰，而新的方法和新的理论不断地产生．

1.1.2　主要特点

计算方法有以下三个主要特点：

第一，面向计算机，提供实际可行的常用算法．具体地说，由于计算机能够进行加、减、乘、除四则运算，故需要把每个求解的数学问题用四则运算的有限形式的公式表达出来．这种公式只能是多项式或有理分式的形式，通常称为算法，它是计算机能够直接处理的．

第二，能够任意逼近，有可靠的理论分析．计算方法有两类算法，一类是精确的，另一类是近似的．所谓精确算法是指在没有运算的舍入误差的假设下，能在确定的运算次数内获得数学问题的精确解．近似算法本身有方法误差，从而在任何有限的运算次数内只能获得数学问题的近似解．大多算法是近似算法．由于计算机字长有限，每次运算都有舍入误差，因此无论精确算法还是近似算法都只能获得数学问题的近似解．计算机需要人们用种种数学理论和方法来建立各个算法．对近似算法要保证收敛性，即近似解能逼近精确解到任意的程度．对每个算法要保证数值稳定性，这是指舍入误差对解的准确性影响不大．

第三，省时间省资源，有良好的计算复杂性．一个算法的计算复杂性是指该算法包含的运算次数和所需的存储量．近似算法的运算次数取决于算法的收敛速度．求解一个数学问题，是否选用或建立计算复杂性好的算法很重要，有时会影响到在现成的计算机上能否真正实现．

从上述特点可以看出，计算方法的任务是提供在计算机上实际可行的、理论可靠的、计算复杂性好的各种算法．

1.1.3　基本线索

函数的插值和逼近是计算方法中最基本的问题之一．目的是提供各种简易的途径，确定函数在一定意义下的近似多项式或近似分段多项式．只要数学问题包含非有理函数，就必须进行函数的数值逼近，才能在计算机上处理．鉴于这个理由，多数数学问题按建立数值方法的基本线索大体上可以

归为两大类.

一类数学问题包含非有理函数或未知函数，如积分与微分计算、微分方程求解等. 这类问题建立数值方法的基本线索是：首先利用函数的数值逼近或离散化将原问题化为近似问题，然后去计算或求解近似问题以得到原问题的近似值或近似解. 例如，对利用各种方式得到的函数的逼近多项式进行求积分或求导数，可以引出各种形式的数值积分或数值微分公式；用一组离散点上待定值代替连续自变量的未知函数，通过各种逼近途径（包括插值、数值积分、数值微分以及 Taylor（泰勒）展开等），将微分方程离散化，构成微分方程的各种数值解法.

另一类数学问题主要是代数问题，包括线性代数方程组的求解和矩阵特征值问题的计算，它们不包含非有理函数，可以直接去建立数值方法而不必先化为近似问题. 非线性（代数的或超越的）方程的求解也可归于此类，因为它们仅仅在求函数值时可能要借助于数值逼近. 这一类问题的数值方法大致可分为直接法、迭代法和变换约化法三种. 直接法是一种精确算法，一般仅用于解线性代数方程组. 迭代法的基本线索是针对确定类型的问题，寻求某种固定形式的递推公式，使得由公式产生的序列收敛于问题的解. 迭代法在计算方法中占有重要的地位，某些数值问题如非线性方程等主要应用迭代法求解. 目前，矩阵特征值问题的大多数重要的数值方法均利用相似变换将矩阵（近似地）约化为特殊形式的矩阵，从而使特征值和特征向量可以较方便地近似求得，这类方法是变换迭代法，可称为变换约化法.

1.2

1.2 误差与有效数字

1.2.1 误差的来源及分类

应用数学解决实际问题时首先需要建立数学模型. 一般

地说，数学模型是指那些利用数学语言模拟现实而建立起来的有关量的描述．数学模型通常总是近似的，其误差称为**模型误差**（Model Error）．

数学模型中常包含某些参量，如比重、温度、电压等，此类量通过观测确定，产生的误差，称为**观测误差**（Observational Error）．

1.2.1 **例** 假设$L(t)$是金属棒在温度为t时的长度，其数学模型为

$l(t)=1+\alpha t+\beta t^2$

其中α、β为参数．有如下估计

$\alpha=0.001253\pm10^{-6}$，$\beta=0.000068\pm10^{-6}$

则$L(t)-l(t)$是模型误差，10^{-6}是α与β的观测误差．

数学模型常常不能获得精确解，必须用数值方法求近似解，其误差称为**截断误差**（Truncation Error）或**方法误差**．

1.2.2 **例** 实际计算时，函数f用Taylor多项式P_n近似代替

$$P_n(x)=f(0)+\frac{f'(0)}{1!}x+\frac{f''(0)}{2!}x^2+\cdots+\frac{f^{(n)}(0)}{n!}x^n$$

则数值方法的截断误差

$$R(x)=f(x)-P_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}x^{n+1}$$

其中，ξ在0与x之间．

有了求解数学问题的算法后，由于计算机的字长有限，原始数据在计算机上表示会产生误差，每一次运算又可能产生新的误差，这种误差称为**舍入误差**（Tounding Error）或**计算误差**．

1.2.3 **例** 实际计算时，用3.14159近似代替π，则舍入误差

$R=\pi-3.14159=0.0000026\cdots$

计算方法中，仅讨论截断误差和舍入误差．

1.2.2 误差概念

1.2.4 **定义** 设 x 为准确值，a 为近似值，记

$$\Delta a = x - a, \qquad \Delta_r a = \frac{\Delta a}{x} = \frac{x-a}{x}$$

称 Δa 为近似值 a 的 **绝对误差**（Absoiute Error）或 **误差**，$\Delta_r a$为a的**相对误差**（Reiative Error）.

因为正是不能取得准确值 x 才去求近似值 a，所以，一般不能算出误差 Δa. 根据测量工具的精度或计算情况的分析，常常能够估计 $|\Delta a|$ 的较好上界. 用 δa 表示这种上界，它是非负数，称为**绝对误差界**. 于是

1.2.5 $|\Delta a| = |x - a| \leqslant \delta a$

这样，就可知道准确值 x 所在的范围：$a - \delta a \leqslant x \leqslant a + \delta a$. 所以，只能说估计误差，而不说计算误差. 在实用上，常用下述写法来刻画 a 的精度 $x = a \pm \delta a$.

由于 x 未知，实际上总是将

1.2.6 $\Delta_r^* a = \dfrac{\Delta a}{a} = \dfrac{x-a}{a}$

作为 a 的相对误差. 记

1.2.7 $\delta_r a = \dfrac{\delta a}{|a|}$

知道绝对误差界 δa 后便确定了 $\delta_r a$，显然，$\delta_r a$ 是 $|\Delta_r^* a|$ 的上界，称为 a 的**相对误差界**.

1.2.3 有效数字

在数值计算时，必须保证各个数据的每一位是可靠的，这是一切有意义的计算的前提. 为此，需要建立有效数字的概念.

当 x 是准确值有很多位数时，常常需要按字长限制取 x

的位数来确定近似值 a，这个 a 按四舍五入规则来选取，能保证绝对误差最小．例如

$x=\pi=3.14159265\cdots$

取 3 位，$a=3.14$，在所有 3 位数中 3.14 与 π 的误差最小；取 5 位，$a=3.1416$，在所有 5 位数中 3.1416 与 π 误差最小．它们的误差均不超过末位的半个单位，即

$$|\pi-3.14|\leqslant\frac{1}{2}\times10^{-2},\quad |\pi-3.1416|\leqslant\frac{1}{2}\times10^{-4}$$

1.2.8 **定义** 设数 x 的近似值

$a=\pm10^{k}\cdot0.a_1a_2\cdots a_n$

其中，每个 $a_i(i=1,2,\cdots,n)$ 是0到 9 中的一个数字，$a_1\neq0$．如果 a 的误差不超过末位的半个单位，即

$$|x-a|\leqslant\frac{1}{2}\times10^{k-n}$$

则称用 a 近似 x 时具有 **n位有效数字**(Significant Figures)．

实际上，有效数字的概念无非是说：按四舍五入规则得到的近似值，其每一位数都是有效的．

显然，有效数字位数与小数点的位置无关．有效位数越多，绝对误差和相对误差就都越小．

有了有效数字概念后，以下写法是有区别的：

34.01,　　34.0100

前者是 4 位有效数字，后者则表示6位有效数字．

1.3

1.3 误差分析

1.3.1 算术运算结果的误差界

1.3.1 **定理** 设 x 与 y 是精确值，a与b是相应的近似值，绝对误差界分别为 δa 与 δb，则

$\delta(a\pm b)=\delta a+\delta b$

$$\delta(ab)\approx|a|\delta b+|b|\delta a$$

$$\delta\left(\frac{a}{b}\right)\approx\frac{|a|\delta b+|b|\delta a}{|b|^2}\quad(b\neq0)$$

1.3.2 例 $\alpha=1.21\times3.65+9.81$，其中每个数据的绝对误差界为 0.005．$\alpha$ 的绝对误差界

$$\begin{aligned}\delta(\alpha)&=\delta(1.21\times3.65)+\delta(9.81)\\&\approx1.21\times0.005+3.65\times0.005+0.005\\&=0.0293\leqslant0.03\end{aligned}$$

1.3.3 定理 设 a 与 b 是近似值，相对误差界分别为 $\delta_r a$ 与 $\delta_r b$，则

$$\delta_r(a+b)=\max\{\delta_r a,\ \delta_r b\}\quad(a与b同号)$$

$$\delta_r(a-b)=\frac{|a|\delta_r a+|b|\delta_r b}{|a-b|}\quad(a与b同号)$$

$$\delta_r(ab)\approx\delta_r a+\delta_r b$$

$$\delta_r\left(\frac{a}{b}\right)\approx\delta_r a+\delta_r b\quad(b\neq0)$$

1.3.4 例 例（1.3.2）中 α 的相对误差界

$$\begin{aligned}\delta_r(\alpha)&=\max\{\delta_r(1.21\times3.65),\ \delta_r(9.81)\}\\&\approx\max\{\delta_r(1.21)+\delta_r(3.65),\ \delta_r(9.81)\}\\&=\max\left\{\frac{\delta(1.21)}{1.21}+\frac{\delta(3.65)}{3.65},\ \frac{\delta(9.81)}{9.81}\right\}\\&=\max\left\{\frac{0.005}{1.21}+\frac{0.005}{3.65},\ \frac{0.005}{9.81}\right\}\\&\approx\max\{0.0055,\ 0.0005\}=0.0055\end{aligned}$$

1.3.2 函数求值的误差估计

在计算一元或多元函数的值时，由于自变量数据不精确会产生误差．

设 f 是一元函数，要计算在点 x 的函数值，但仅知 x 的

近似值为 a，以$f(a)$近似$f(x)$. $f(a)$的绝对误差界$\delta(f(a))$可用 Taylor 公式估计. 假定f在包含x与a的一个开区间上存在足够阶的导数，则有

$$\Delta(f(a))=f(x)-f(a)=f'(a)(x-a)+\frac{f''(\xi)}{2!}(x-a)^2$$

其中ξ在x与a之间. 取绝对值，得

$$|\Delta(f(a))|=|f(x)-f(a)|\leqslant|f'(a)|\,\delta a+\frac{|f''(\xi)|}{2}(\delta a)^2$$

假定$|f''(a)|$与$|f'(a)|$相比不太大，则可以忽略高阶项，得

1.3.5 $\delta(f(a))\approx|f'(a)|\delta a$

但是，如果$|f'(a)|$是零或值很小，则要考虑后面的项. 特别地，若

$f'(a)=f''(a)=\cdots=f^{(k-1)}(a)=0$，$f^{(k)}(a)\neq 0$

且$|f^{(k+1)}(\xi)|$（ξ在x与a之间任意变动）不很大，则

1.3.6 $\delta(f(a))\approx\dfrac{|f^{(k)}(a)|}{k!}(\delta a)^k$

1.3.7 **例** 设$f(x)=\sin x$，$\alpha=45°$，$\beta=90°$，$\delta\alpha=0.1°$，$\delta\beta=0.2°$. 由于

$$f'(\alpha)=\cos_\alpha=\frac{\sqrt{2}}{2},\quad f'(\beta)=0,\quad f''(\beta)=-\sin\beta=-1$$

所以，在把$\delta\alpha$与$\delta\beta$化为弧度后有

$$\delta(\sin\alpha)\approx|\cos\alpha|\,\delta\alpha=\frac{\sqrt{2}}{2}\times 0.1\times\frac{\pi}{180}\approx 1.2\times 10^{-3}$$

$$\delta(\sin\beta)\approx\frac{|\sin\beta|}{2!}(\delta\beta)^2=\frac{1}{2}\times\left(0.2\times\frac{\pi}{180}\right)^2\approx 6.1\times 10^{-6}$$

多元函数值的误差界可用多元函数的Taylor公式得到

1.3.8 $\delta\Big(f(a_1,a_2,\cdots,a_n)\Big)\approx\sum\limits_{i=1}^{n}\left|\dfrac{\partial f(a_1,a_2\cdots,a_n)}{\partial x_i}\right|\delta a_i$

如果一阶偏导数绝对值都很小或等于零，则要利用高阶项，其一般描述与一元函数类似．

1.3.9 例 设$f(x,y)=xy$，a和b分别是x和y的近似值，则

$$\delta(f(a,b))\approx|f_x(a,b)|\delta a+|f_y(a,b)|\delta b=|b|\delta a+|a|\delta b$$

1.3.3 误差分析的方法

误差分析是计算方法中一个重要而复杂的问题．前面列举的简单的情形，提供了不精确数据运算结果的误差界，它们都假定运算可以精确进行．然而，对工程上或科学上的数值计算，误差分析则要困难得多．如果算法是近似的，就需进行截断误差分析，而且运算次数往往数以千万次计．原始数据有误差，每一步运算会产生新的舍入误差并传播前面各步已引入的误差，所以按步分析误差自然是办不到的．总的效果是有正有负、绝对值有大有小的各种误差的积累，人们关心的是估计积累误差的界．解决这个问题似乎尚无统一的理论，目前，积累误差的界是针对不同的问题逐例进行分析的，而且以截断误差与舍入误差的分别研究为基础．

关于误差界有两个术语：假设在某些条件下推导出一个理论上的误差界，而这个界不依赖于计算结果，则称它为**先验界**或**先验估计**(Prior Estimate)；如果这个界依赖于计算结果，也就是说，仅在计算完成后利用计算的结果才能具体确定这个界，则称它为**后验界**或**后验估计**（Posterior Estimate)．例如，设某个递推公式理论上确定的序列为x_1，x_2，…，而实际算得的序列为a_1，a_2，…，并且按问题的条件能够得到估计

$$|x_k-a_k|\leqslant\sum_{i=1}^{k-1}A_i|a_i|\qquad(k=1,2,\cdots)$$

其中A_i是确定的非负常数．这个公式给出的界$\sum\limits_{i=1}^{k-1}A_i|a_i|$依

赖于 $a_1, a_2\cdots, a_{k-1}$，它们在计算之前是未知的，故仅能作为后验界. 若经推导 a_i 还满足条件

$$|a_i| \leqslant c \quad (i=1,2,\cdots)$$

其中 c 是确定的正常数，则能得到

$$|x_k - a_k| \leqslant c \sum_{i=1}^{k-1} A_i \qquad (k=1,2,\cdots)$$

这个 $c\sum_{i=1}^{k-1} A_i$ 就是先验界.

针对不同问题已经建立起一些误差分析法，目的是提供误差的某个先验界或后验界. 对于舍入误差积累问题，除了估计误差界外，还需有数值稳定性的概念. 一个数值稳定的算法是指在执行它的过程中舍入误差在一定条件下能够得到控制；数值不稳定的算法不能实际使用. 数值稳定性的分析，特别对大运算量或运算次数不受限制的问题有重要意义. 关于稳定性分析还有一些特殊的方法，可参见微分方程数值解法的有关章节.

常用的误差分析法有以下几种：

向前误差分析法与向后误差分析法是分析算法舍入误差积累（不涉及截断误差）的两种不同方法. 假设所讨论的算法由若干公式表达，某个新的量 x 由已知量（前面已算出的量或原始数据）$a_1, a_2\cdots, a_n$ 的某个公式定义，写成

$$x = g(a_1, a_2, \cdots, a_n)$$

x 从 $a_1, a_2, \cdots, a_n$ 经过基本的算术运算得出. 因为计算中产生舍入误差，实际计算值记作 x_{fl}（用 fl 表示是由计算机浮点计算得到的），它与精确值 x 不同. **向前误差分析**（Forward Error Analysis）是对每一步计算找出舍入误差界，随计算过程逐步向前分析，直至估计出最后结果的舍入误差 $|x - x_{fl}|$ 的界. **向后误差分析**（Backward Error Analysis）则是

把舍入误差与导出 x_{fl} 的已知量 a_1，$a_2,\cdots,a_n$ 的某种摄动（误差）等价起来，即对每个 a_i 引进某个摄动量 ε_i，使得精确地成立

$$x_{fl}=g(a_1+\varepsilon_1,\ a_2+\varepsilon_2,\cdots,a_n+\varepsilon_n)$$

并推出这些 ε_i 的界（并非要得出 ε_i 的具体值，ε_i 不是唯一的），然后利用摄动理论估计最后的舍入误差界。向后误差分析法是一种先验估计法，特别在数值线代数（矩阵运算）的误差研究中有比较系统的应用，取得了较大的进展．相比之下，向前误差分析法只能应用于十分简单的情形．

区间分析法是出现不久的一种研究误差的方法，它主要利用区间分析这一数学新分支中的区间运算理论．设 x,y 是准确值，α，β 是相应的近似值，且已知绝对误差界 $\delta\alpha,\delta\beta$，则能确定 x，y 的所在区间

$$\alpha-\delta\alpha\leqslant x\leqslant\alpha+\delta\alpha,\quad \beta-\delta\beta\leqslant x\leqslant\beta+\delta\beta$$

或者写成

$$x\in[\alpha-\delta\alpha,\ \alpha+\delta\alpha],\quad y\in[\beta-\delta\beta,\ \beta+\delta\beta]$$

这样，利用 x,y 的所在区间，按照区间分析中的区间运算，能够得出 x 与 y 之间各种运算的精确结果的所在区间，并由这个区间给出实际运算结果的误差估计，实际运算自然是在 α 与 β 之间进行的．这就是区间分析法的基本思想．

前面提供的关于不精确数据运算结果的误差界，及舍入误差分析引出的误差界，通常远远大于实际的误差．实际上误差分布有随机性，不会经常地达到上界．因此，利用概率和统计方法，将数据和运算中的误差视为适合某种分布的随机变量，然后确定计算结果的误差分布，并用它代替绝对误差界，常常可使误差估计更接近实际，这种误差分布称为**概率界**，这种分析方法就是**概率分析法**．

1.4

1.4 数值运算的一些简单原则

关于数值运算的若干简单原则是减少舍入误差影响的一些普通常识，有以下几点：

1° 注意运算次序

在计算机中，由于字长的限制，$(a+b)+c$ 可能不等于 $a+(b+c)$，因为两者的舍入误差可能不同.例如，以限制取两位十进制数为例，$(10^1\cdot 0.19+10^{-1}\cdot 0.43)+10^{-1}\cdot 0.47$ 的两位近似值为 $10^1\cdot 0.19$，舍入误差为 $10^{-1}\cdot 0.90$；而 $10^1\cdot 0.19+(10^{-1}\cdot 0.43+10^{-1}\cdot 0.47)$ 的两位近似值为 $10^1\cdot 0.20$，舍入误差为 $10^{-1}\cdot 0.10$. 由此例可以看出一个简单原则：在多个数求和时，如果被加数的绝对值之间差异较大，且包含许多绝对值较小的数，则应按绝对值从小到大的次序相加.

1.4.1 **例** 在4位十进制的限制下，计算

$A=1000+\delta_1+\delta_2+\cdots+\delta_{1000}$

其中 $0.1\leqslant\delta_i\leqslant 0.4$，$i=1,2,\cdots,1000$.

解 如果自左到右相加，则每个 δ_i 被 1000 "吃掉"，A 的 4 位近似值为1000；如果先把所有 δ_i 加起来再加 1000，则

$1100=1000+0.1\times 1000\leqslant A\leqslant 1000+0.4\times 1000=1400$

但是，这种运算次序的原则并不是绝对的，在实际计算中，存在着比较特殊的相反例子，它们按绝对值从大到小的次序相加，其舍入误差反而小.

1.4.2 **例** 在4位十进制的限制下，计算

$10^4\cdot 0.1025+(-10^3\cdot 0.9123)+(-10^2\ 0.9663)$

解 按自左到右次序相加得 $10^2\cdot 0.1607$，没有舍入误差，按自右到左次序相加得到 4 位近似值 $10^2\cdot 0.1600$，舍入误差为0.07.

2° 尽量避免相近数相减

相近数相减会使有效数字大量丢失，如有可能应尽量避

免．主要是在构造具体算法时要周密地考虑，防止产生这样的情况．

1.4.3 例 二次方程$ax^2+bx+c=0$ 的根的公式通常形式为

$$x_1=\frac{-b+\sqrt{b^2-4ac}}{2a},\quad x_2=\frac{-b-\sqrt{b^2-4ac}}{2a}$$

当遇到$b^2\gg 4|ac|$的情形时，有$|b|\approx\sqrt{b^2-4ac}$，用上述公式求出的两个根中，总有一个因相近数相减而严重不可靠．为了求得可靠的结果，鉴于$x_1x_2=c/a$，在计算机上采用如下算法：

$$x_1=\frac{-b-\operatorname{sign}(b)\sqrt{b^2-4ac}}{2a},\quad x_2=\frac{c}{ax_1}$$

这里的x_1与上面的x_1，当$b<0$时是相同的．这个公式避免了相近数相减的可能性．

3° 避免被除数绝对值远远大于除数绝对值的除法

绝对值大的数被绝对值小的数除，舍入误差界要比相反的情形大，有时会给计算结果带来严重的影响．

1.4.4 例 求解二元线性方程组

$$\begin{cases}0.0001x_1+x_2=1\\ x_1+x_2=2\end{cases}$$

解 此方程组的精确解为

$$x_1=\frac{10000}{9999},\quad x_2=\frac{9998}{9999}$$

现在，在3位十进制的限制下用消去法求解，上述方程组应改写成

$$\begin{cases}10^{-3}\cdot 0.100x_1+10^1\cdot 0.100x_2=10^1\cdot 0.100\\ 10^1\cdot 0.100x_1+10^1\cdot 0.100x_2=10^1\cdot 0.200\end{cases}$$

若利用第一个方程消去第二个方程中含x_1的项，将第二

个方程减去第一个方程的 $(10^{-3}\cdot 0.100)^{-1}$倍，则出现大数被小数除的情形，得到

$$\begin{cases} 10^{-3}\cdot 0.100x_1 + 10^1\cdot 0.100x_2 = 10^1\cdot 0.100 \\ \qquad\qquad - 10^5\cdot 0.100x_2 = -10^5\cdot 0.100 \end{cases}$$

由此解出

$x_1 = 0$（失去近似意义），$x_2 = 10^1\cdot 0.100$

若反过来用第二个方程消去第一个方程中含x_1的项，则避免出现大数被小数除的情形，得到

$$\begin{cases} \qquad\qquad 10^1\cdot 0.100x_2 = 10^1\cdot 0.100 \\ 10^1\cdot 0.100x_1 + 10^1\cdot 0.100x_2 = 10^1\cdot 0.200 \end{cases}$$

由此得出相当好的近似解

$x_1 = x_2 = 10^1\cdot 0.100$

4° 简化计算步骤

每次算术运算都可能产生舍入误差，因此，如能通过算法的改进减少运算次数，特别是减少乘除法的运算次数，则舍入误差的积累一般可能下降，还能节省计算机的执行时间。

算法是通过在具体执行上没有任何随意性的表达式来描述的计算过程，因此不是任何一个表达式都确定一个算法。

1.4.5 例 求多项式

$P_n(x) = a_n x^n + a_{n-1}x^{n-1} + \cdots + a_1 x + a_0$

的值。

解 多项式是一个表达式，但由它导出结果的过程有很大的随意性，它不是一个算法而是一个计算问题。若直接计算$a_k x^k$再逐项相加，则确定一种算法：

$$\begin{cases} A_0 = a_0 \\ A_k = a_k x^k, \qquad k = 1, 2, \cdots n \\ P_n(x) = A_0 + A_1 + \cdots + A_n \end{cases}$$

这个算法有明显的缺点，比如x的幂的重复计算，它共需做

$$1+2+\cdots+n=\frac{n(n+1)}{2}$$

次乘法和 n 次加法。

1.4.6 著名的秦九韶算法或称Horner算法如下：

$$\begin{cases} S_n=a_n \\ S_k=xS_{k+1}+a_k \qquad (k=n-1,\cdots,2,1,0) \\ P_n(x)=S_0 \end{cases}$$

利用此算法求 $P_n(x)$ 的值只须做 n 次乘法和 n 次加法。

1.4.7 例 $\frac{2}{3}=\frac{1}{3}\cdot 2$，在4位十进制的限制下，$\frac{2}{3}$的近似值为 $10^0\cdot 0.6667$，舍入误差为 $10^{-4}\cdot 0.44\cdots$；$\frac{1}{3}\cdot 2$比$\frac{2}{3}$多一次运算，近似值为 $10^0\cdot 0.6666$，舍入误差为$10^{-4}\cdot 0.66\cdots$。

2

第2章 插值法

2.1

2.1 引 言

2.1.1 插值的意义

函数插值的提出源于关于函数的两个方面的问题。一方面，在实际问题中出现的函数，常有赖于实验和观察，虽然可能在某个区间上有定义，却没有明确的表达式，而只能得到区间内一些离散点上的值，因此希望对这样的函数能用简单表达式近似地给出整体上的描述，并能与已知的离散点上的值相符。另一方面，函数有明确的表达式，但只要不是（分段）有理函数，便不易计算和使用，这样，也需要用简单的函数提供一个好的逼近。

在插值的研究工作中，对用于逼近的简单函数的类型有不同的选取。多项式或分段多项式最便于计算和使用，因而最引人注目。特别是计算机出现后，人们的注意力更集中在利用多项式的插值方面，这是因为计算公式相对地易于描述和进行程序设计，其误差分析也比较简单。插值还是建立适用于计算机的许多其它算法的一个基本工具。

2.1.2 插值问题的提法

设 f 是区间$[a,b]$上的一个实函数，且已知离散数据

2.1.1 (x_i,y_i)，$y_i=f(x_i)$ $(i=0,1,\cdots,n)$

其中x_0，$x_1,\cdots,x_n$是$[a,b]$上包含 a 和 b 在内的 $n+1$ 个相异的实数。

插值问题最基本的提法是：寻求一个次数尽可能低的多

项式 p，满足条件

2.1.2 $p(x_i)=y_i \ (i=0,1,\cdots,n)$

从几何上看，就是寻求一个最低次的多项式，其几何曲线通过给定的 $n+1$ 个点 (x_i,y_i)，$(i=0,1,\cdots,n)$.

如果所说的多项式 p 存在，并用它近似函数f(p 与 f 有 $n+1$ 个相同的值)，则称 p 为 f 的**插值多项式**（Interpolation Polynomial），$x_0, x_1, \cdots, x_n$ 称为插值节点或简称节点(Node)，$[a,b]$ 称为**插值区间**，条件(2.1.2)称为**插值条件**，f称为**被插函数**（Interpolated Function）.

插值问题有进一步的提法．一种提法是寻求满足插值条件(2.1.2)的分段多项式；另一种提法是插值条件中增加在某些节点上的导数值的条件，寻求相应的多项式．

2.1.3 插值多项式的存在唯一性

2.1.3 **定理** 存在唯一的、次数不超过 n 的多项式 p，使得满足插值条件 $p(x_i)=y_i$，$i=0,1,\cdots,n$.

此定理有不同的证法．可通过构造多项式证明存在性，利用多项式的次数与根的个数的关系证明唯一性．也可假设

2.1.4 $p(x)=a_nx^n+a_{n-1}x^{n-1}+\cdots+a_1x+a_0$

且满足插值条件，即

2.1.5 $a_nx_i^n+a_{n-1}x_i^{n-1}+\cdots+a_1x_i+a_0=y_i \ (i=0,1,\cdots,n)$

这是关于 $a_0, a_1, \cdots, a_n$的线性代数方程组，因系数行列式不为零，故有唯一解．从而 p 是存在唯一的．

插值问题的关键，是怎样具体去求插值多项式．求插值多项式的方法称为**插值法**（Method of Interpolation）.

利用解方程组（2.1.5）去建立形如（2.1.4）的插值多项式，计算量大，有时还会对精度有较大的影响，因而是不可取的．插值法是一些比较方便的方法，它们对插值多项式的形式作了各种特殊的选取，以便较容易地去具体确定．

2.1.4 插值法的主要线索

插值法是一个古老的课题。早在公元6世纪，我国刘焯已将等距二次插值应用于天文计算。17世纪，Newton（牛顿）和Gregory（格雷哥里）建立了等距节点上的一般插值公式。18世纪，Lagrange（拉格朗日）给出了更一般的非等距节点上的插值公式。由于应用上和理论上的需要，近几十年来，插值法仍不断有新的发展。

插值法的主要线索是：

1° 非等距节点插值包括Lagrange插值，Aitken（埃特金）插值，利用均差的Newton插值。

2° 等距节点插值。包括利用差分的Newton公式，以及Gauss（高斯）公式等若干不同形式的插值。

3° Hermite（埃尔米特）插值。这是插值条件中增加了导数值条件的插值。

4° 分段低次插值。包括分段线性插值，分段Hermite插值，特别还有近年发展起来的利用样条函数的插值，它已获得了广泛的应用。

5° 反插值。

2.2 Lagrange插值

2.2.1 基函数

考虑最简单的插值问题，设离散数据为$\{(x_k,\delta_{ik})\}_{k=0}^{n}$，这里$i$是一个非负整数，$0\leqslant i\leqslant n$，$\delta_{ik}$是Kronecker（克隆内克）符号

$$\delta_{ik}=\begin{cases}1 & (k=i)\\ 0 & (k\neq i)\end{cases} \tag{2.2.1}$$

求插值多项式，记这个多项式为l_i。

由于多项式l_i必须满足

$$l_i(x_k)=\delta_{ik}\quad(k=0,1,\cdots,n) \tag{2.2.2}$$

即 $x_0,\cdots,x_{i-1}$, $x_{i+1},\cdots$, x_n 是l_i的根，因此可取如下形式

$$l_i(x)=a(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)$$

而且从条件 $l_i(x_i)=1$ 可确定出系数 a，最后得出

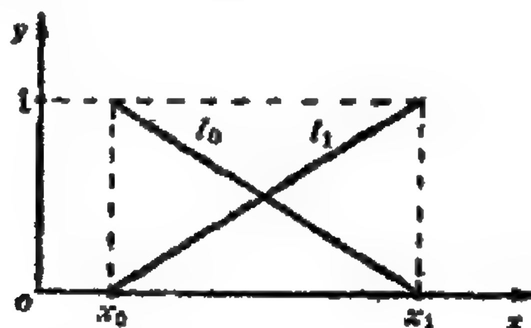
2.2.3 $$l_i(x)=\frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

$$=\prod_{\substack{j=0\\ j\neq i}}^{n}\frac{x-x_j}{x_i-x_j}$$

$n+1$ 个 n 次多项式 l_0，$l_1,\cdots,l_n$称为多项式插值的以x_0，$x_1,\cdots$，x_n 为节点的 n 次**基函数** (Basis Function)。

2.2.4 $n=1$时的基函数及图形

$$l_0(x)=\frac{x-x_1}{x_0-x_1},\qquad l_1(x)=\frac{x-x_0}{x_1-x_0}$$

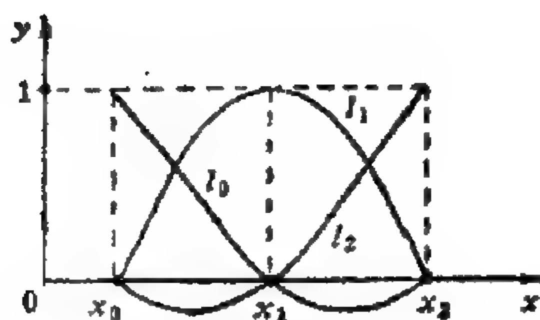


2.2.5 $n=2$时的基函数及图形

$$l_0(x)=\frac{(x-x_1)(x-x_2)}{(x_0-x_1)(x_0-x_2)}$$

$$l_1(x)=\frac{(x-x_0)(x-x_2)}{(x_1-x_0)(x_1-x_2)}$$

$$l_2(x)=\frac{(x-x_0)(x-x_1)}{(x_2-x_0)(x_2-x_1)}$$



2.2.2 Lagrange插值多项式

一般离散数据 $\{(x_i,y_i)\}_{i=0}^n$ 的插值多项式 L_n是基函数 l_0, $l_1,\cdots,l_n$的线性组合：

2.2.6 $$L_n(x)=\sum_{i=0}^{n} y_i l_i(x)$$

$$=\sum_{i=0}^{n} y_i \frac{(x-x_0)\cdots(x-x_{i-1})(x-x_{i+1})\cdots(x-x_n)}{(x_i-x_0)\cdots(x_i-x_{i-1})(x_i-x_{i+1})\cdots(x_i-x_n)}$$

显然，L_n满足插值条件，即

$L_n(x_j)=y_j \quad (j=0,1,\cdots,n)$

L_n称为 Lagrange**插值多项式**。记

2.2.7 $$W_{n+1}(x)=\prod_{i=0}^{n}(x-x_i)=(x-x_0)(x-x_1)\cdots(x-x_n)$$

于是 (2.2.6) 可改写成

2.2.8 $$L_n(x)=W_{n+1}(x)\sum_{i=0}^{n}\frac{y_i}{(x-x_i)W'_{n+1}(x_i)}$$

$$(x\neq x_i, i=0,1,\cdots,n)$$

2.2.9 **例** 设有离散数据

(1,0)，(2,−5)，(3,−6)，(4,3)

则插值多项式为

$$L_3(x)=0\cdot\frac{(x-2)(x-3)(x-4)}{(1-2)(1-3)(1-4)}+(-5)\cdot\frac{(x-1)(x-3)(x-4)}{(2-1)(2-3)(2-4)}$$

$$+(-6)\cdot\frac{(x-1)(x-2)(x-4)}{(3-1)(3-2)(3-4)}$$

$$+3\cdot\frac{(x-1)(x-2)(x-3)}{(4-1)(4-2)(4-3)}$$

$$=x^3-4x^2+3$$

2.2.3 余项

设在区间$[a,b]$上用插值多项式L_n逼近函数f，节点$x_0, x_1,\cdots,x_n\in[a,b]$，且$f(x_i)=y_i$，$i=0,1,\cdots,n$，则其截断误差 R_n是$[a,b]$上的函数

2.2.10 $R_n(x)=f(x)-L_n(x)$

R_n也称为L_n的**余项**(Remainder Term)。这时，有下面的余项估计定理。

2.2.11 **定理** 若 f在$[a,b]$上存在 $n+1$ 阶导数，则对任何 $x\in[a,b]$有

2.2.12 $R_n(x)=f(x)-L_n(x)=\dfrac{f^{(n+1)}(\xi)}{(n+1)!}W_{n+1}(x)$

其中 $\xi\in[a,b]$且依赖于x，W_{n+1}由（2.2.7）定义。

ξ一般无法具体确定，因此（2.2.12）只是一个估计式。如果$f^{(n+1)}$在$[a,b]$上有界（或连续），即存在常数 $M_{n+1}>0$（连续时可取 $M_{n+1}=\max\limits_{a\leqslant x\leqslant b}|f^{(n+1)}(x)|$）

$|f^{(n+1)}(x)|\leqslant M_{n+1}$ ，$\forall x\in[a,b]$

则有余项估计

2.2.13 $|R_n(x)|\leqslant\dfrac{M_{n+1}}{(n+1)!}|W_{n+1}(x)|$

对于 $n=1$ 的情况，L_1称为**线性插值多项式**. 设节点 $x_0<x_1$，函数f在区间$[x_0,x_1]$上有连续二阶导数，$y_0=f(x_0)$，$y_1=f(x_1)$。在$[x_0,x_1]$上用 L_1逼近f，

2.2.14 $$L_1(x)=y_0\frac{x-x_1}{x_0-x_1}+y_1\frac{x-x_0}{x_1-x_0}$$

其余项为

$$R_1(x)=\frac{1}{2}f''(\xi)W_2(x)$$

$$=\frac{1}{2}f''(\xi)(x-x_0)(x-x_1),\quad \xi\in[x_0,x_1]$$

由于 $|W_2(x)|$ 在 $x=\frac{x_0+x_1}{2}$达最大值，可得余项的一个界

2.2.15 $$|R_1(x)|\leqslant\frac{1}{2}M_2\left|W_2\left(\frac{x_0+x_1}{2}\right)\right|$$

$$=\frac{M_2(x_1-x_0)^2}{8}$$

其中 $M_2=\max\limits_{x_0\leqslant x\leqslant x_1}|f''(x)|$

若是为了建立插值多项式，则只要知道函数 f在节点（离散点）上的值就足够了；但如果要给出余项估计，则要求f在一个区间上不仅有明确的表达式，而且还存在高阶导数。

2.3

2.3 Aitken法

2.3.1 问题的提出

Lagrange 插值多项式的优点是简单、易于建立。缺点是增加新节点时原有多项式的计算结果不能利用，必须重新建立；而且其形式不易简化，计算工作量较大。

为了克服上述两个缺点，产生了一些新的途径。Aitken法利用一列 Lagrange 插值多项式，避免了增加新节点时从

头开始计算.Newton 插值则从改进插值多项式的形式入手，以便于增加节点和节省计算工作量.

2.3.2 Aitken 法的描述

用 $p_{n_0,n_1,\cdots,n_k}$ 表示节点为 $x_{n_0},x_{n_1},\cdots,x_{n_k}$ 的Lagrange插值多项式，p_{n_i} 是零次多项式即为常数，

2.3.1 $p_{n_i}(x)=f(x_{n_i})$

这里 $n_0,n_1,\cdots,n_k$ 是非负整数，f 是被逼近的函数。

容易证明（只要验证满足插值条件）

2.3.2 $$p_{0,1,\cdots,k,n}(x)=\frac{1}{x_n-x_k}\begin{vmatrix} p_{0,1,\cdots,k}(x) & x_k-x \\ p_{0,1,\cdots,k-1,n}(x) & x_n-x \end{vmatrix}$$

$$=p_{0,1,\cdots,k}(x)\frac{x-x_n}{x_k-x_n}+p_{0,1,\cdots,k-1,n}(x)\frac{x-x_k}{x_n-x_k}$$

公式（2.3.2）表明，$k+1$ 次插值多项式可由两个 k 次插值多项式通过线性插值得到。因此，利用这个公式，可以一行接一行地构造表（2.3.3）。例如，

第 2 行中

$$p_{0,1}(x)=\frac{1}{x_1-x_0}\begin{vmatrix} p_0(x) & x_0-x \\ p_1(x) & x_1-x \end{vmatrix}$$

$$p_{0,2}(x)=\frac{1}{x_2-x_0}\begin{vmatrix} p_0(x) & x_0-x \\ p_2(x) & x_2-x \end{vmatrix}$$

第 3 行中

$$p_{0,1,2}(x)=\frac{1}{x_2-x_1}\begin{vmatrix} p_{0,1}(x) & x_1-x \\ p_{0,2}(x) & x_2-x \end{vmatrix}$$

等等。表中第 $n+1$ 行的最后一个多项式即为 n 次插值多项式。整个计算过程称之为 Aitken 法。

2.3.3 表

已知数据		按(2.3.2)构造多项式			
x_0	$p_0(x)=f(x_0)$				
x_1	$p_1(x)=f(x_1)$	$p_{0,1}(x)$			
x_2	$p_2(x)=f(x_2)$	$p_{0,2}(x)$	$p_{0,1,2}(x)$		
x_3	$p_3(x)=f(x_3)$	$p_{0,3}(x)$	$p_{0,1,3}(x)$	$p_{0,1,2,3}(x)$	$\ddots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	
x_n	$p_n(x)=f(x_n)$	$p_{0,n}(x)$	$p_{0,1,n}(x)$	$p_{0,1,2,n}(x)$	$\cdots p_{0,1,\cdots,n}(x)$

利用 Aitken 法，每增加一个节点，只要在表 (2.3.3) 中多计算一行即可。

2.3.3 计算工作量

比较在求 n 次插值多项式的值时， Lagrange 插值与 Aitken 法的运算次数。

通过计算，先将表达式 (2.2.6) 化为

$$L_n(x)=\sum_{i=0}^{n}\left(a_i\prod_{\substack{j=0\\ j\neq i}}^{n}(x-x_j)\right) \tag{2.3.4}$$

计算

$$a_i=y_i\Big/\prod_{\substack{j=0\\ j\neq i}}^{n}(x_i-x_j)\quad(i=0,1,\cdots,n)$$

需 n^2-1 次乘法， $n+1$ 次除法， n^2+n 次减法. 然后利用 (2.3.4)式求值，每次求值需n^2+n次乘法，n^2+n次减法，n次加法. 如果仅求一个值，Lagrange 插值的计算工作量总共为 $2n^2+n-1$ 次乘法，$n+1$ 次除法，$2n^2+2n$ 次减法及 n 次加法.

Aitken 法求 n 次插值多项式的值可以直接从表(2.3.3)

出发，表中共有 $n(n+1)/2$ 项，每项按公式(2.3.2) 求值需 2 次乘法，1 次除法及 4 次减法，因此求 n次多项式的一个值共需 n^2+n 次乘法，$n(n+1)/2$ 次除法，$2n^2+2n$次减法。

2.3.5 例 已知 $f(x)=\sinh x$ 的离散数据

x_i	0.00	0.20	0.30	0.50
$f(x_i)$	0.00000	0.20134	0.30452	0.52110

利用三次插值求 $f(0.23)$的近似值。

解 采用 Lagrange 插值

$$f(0.23)\approx 0.20134\cdot\frac{0.23\cdot(-0.07)\cdot(-0.27)}{0.20\cdot(-0.10)\cdot(-0.30)}$$

$$+0.30452\cdot\frac{0.23\cdot 0.03\cdot(-0.27)}{0.30\cdot 0.10\cdot(-0.20)}$$

$$+0.52110\cdot\frac{0.23\cdot 0.03\cdot(-0.07)}{0.50\cdot 0.30\cdot 0.20}$$

$$\approx 0.232035$$

采用 Aitken 法列表如下：

0.00	0.00000				−23
0.20	0.20134	0.231541			−3
0.30	0.30452	0.233465	0.232118		7
0.50	0.52110	0.239706	0.232358	<u>0.232034</u>	27

如果增加一个节点，例如 $x_4=0.60$，其函数值 $f(0.60)=0.63665$，并改用四次插值计算$f(0.23)$的近似值。则Lagrange 插值必须从头计算，而Aitken法只增加如下一行：

0.60	0.63665	0.244049	0.232479	0.232034	<u>0.232034</u>	37

2.4

2.4 均差与Newton插值

2.4.1 均差

设函数 f 在 $n+1$ 个节点 $x_0, x_1, \cdots, x_n$ 上的值 $f(x_0)$, $f(x_1), \cdots, f(x_n)$为已知. Newton 插值是把插值多项式表示成如下形式:

2.4.1 $$N_n(x)=a_0+a_1(x-x_0)+a_2(x-x_0)(x-x_1)+\cdots+a_n(x-x_0)(x-x_1)\cdots(x-x_{n-1})$$

这种形式显然便于求值，而且增加一个新节点 x_{n+1} 时，只须增加一个新项

$$a_{n+1}(x-x_0)(x-x_1)\cdots(x-x_n)$$

根据插值条件

$$N_n(x_i)=f(x_i)\quad(i=0,1,\cdots,n)$$

可以逐个确定出系数

$$a_0=f(x_0)$$

$$a_1=\frac{f(x_1)-f(x_0)}{x_1-x_0}$$

$$a_2=\frac{\dfrac{f(x_2)-f(x_0)}{x_2-x_0}-\dfrac{f(x_1)-f(x_0)}{x_1-x_0}}{x_2-x_1}$$

$\cdots$

为了得到系数 a_i 的一般表达式，有如下均差的定义.

2.4.2 **定义** 记 $f[x_m]=f(x_m)$,称为**零阶均差**.零阶均差的差商

$$f[x_0,\ x_m]=\frac{f(x_m)-f(x_0)}{x_m-x_0}$$

称为函数 f 关于 x_0, x_m 的**一阶均差**. 一阶均差的差商

$$f[x_0, x_1, x_m]=\frac{f[x_0, x_m]-f[x_0, x_1]}{x_m-x_1}$$

称为 f 的**二阶均差**. 一般地，$k-1$ 阶均差的差商

2.4.3 $f[x_0, x_1, \cdots, x_{k-1}, x_m]$

$$= \frac{f[x_0, \cdots, x_{k-2}, x_m] - f[x_0, x_1, \cdots, x_{k-1}]}{x_m - x_{k-1}}$$

称为 f 的 k 阶均差(K－th Divided Difference)。

均差有如下基本性质：

1° 均差与函数值的关系为

2.4.4 $f[x_0, x_1, \cdots, x_n]$

$$= \sum_{j=0}^{n} \frac{f(x_j)}{(x_j - x_0)\cdots(x_j - x_{j-1})(x_j - x_{j+1})\cdots(x_j - x_n)}$$

可用归纳法给以证明.

2° 均差关于所含节点是对称的，即有

$f[x_0, x_1, \cdots, x_n]$

$= f[x_1, x_0, x_2, \cdots, x_n] = \cdots = f[x_1, \cdots, x_n, x_0]$

这是1°的直接推论.

3° 由2°及（2.4.3)可得

2.4.5 $f[x_0, x_1, \cdots, x_{k-1}, x_m]$

$$= \frac{f[x_1, \cdots, x_{k-1}, x_m] - f[x_0, x_1, \cdots, x_{k-1}]}{x_m - x_0}$$

4° 设 f 在$[a,b]$上存在 n 阶导数，且$x_0, x_1, \cdots, x_n \in [a,b]$，则

2.4.6 $f[x_0, x_1, \cdots, x_n] = \frac{f^{(n)}(\xi)}{n!}$

其中 $\xi \in [a,b]$。

2.4.2 Newton插值公式

利用（2.4.5)，可以得到

$f(x) = f(x_0) + f[x, x_0](x - x_0)$

$$f[x, x_0] = f[x_0, x_1] + f[x, x_0, x_1](x - x_1)$$

$$f[x, x_0, x_1] = f[x_0, x_1, x_2] + f[x, x_0, x_1, x_2](x - x_2)$$

$$\cdots$$

$$f[x, x_0, \cdots, x_{n-1}] = f[x_0, x_1, \cdots, x_n] + f[x, x_0, \cdots, x_n](x - x_n)$$

依次将后一式代入前一式，最后有

2.4.7 $f(x) = N_n(x) + R_n(x)$

其中

2.4.8 $N_n(x) = f(x_0) + f[x_0, x_1]\ (x - x_0)$

$$+ f[x_0, x_1, x_2]\ (x - x_0)(x - x_1) + \cdots$$

$$+ f[x_0, x_1, \cdots, x_n](x - x_0)(x - x_1)\cdots(x - x_{n-1})$$

2.4.9 $R_n(x) = f(x) - N_n(x) = f[x, x_0, \cdots, x_n]W_{n+1}(x)$

W_{n+1} 由 (2.2.7) 定义.

由 (2.4.8) 确定的多项式 N_n 显然满足插值条件，次数不超过 n，且具有 (2.4.1) 的形式. 因此，(2.4.1) 式中的系数

$$a_i = f[x_0, x_1, \cdots, x_i] \quad (i = 0, 1, \cdots, n)$$

多项式 N_n 称为 Newton**均差插值多项式**. 根据插值多项式的唯一性，N_n 与 Lagrange 插值多项式 L_n 只是形式上不同.

当 f 存在 $n+1$ 阶导数时，公式 (2.4.9) 给出均差形式的余项 R_n 与公式 (2.2.12) 等价，并由此可推出均差与导数的关系式 (2.4.6)。但公式 (2.4.9) 更有一般性，它在 f 是由离散数据确定或导数不存在的情形下仍有意义.

2.4.3 计算工作量

计算均差可按表 (2.4.10) 逐行地进行，表中加横线的各阶均差就是 Newton 插值多项式的系数.

2.4.10 表

x_i	$f(x_i)$	一阶均差	二阶均差	三阶均差	…	n阶均差
x_0	$\underline{f(x_0)}$					
x_1	$f(x_1)$	$\underline{f[x_0,x_1]}$				
x_2	$f(x_2)$	$f[x_0,x_2]$	$\underline{f[x_0,x_1,x_2]}$			
x_3	$f(x_3)$	$f[x_0,x_3]$	$f[x_0,x_1,x_3]$	$\underline{f[x_0,x_1,x_2,x_3]}$		
⋮	⋮	⋮	⋮	⋮	⋱	
x_n	$f(x_n)$	$f[x_0,x_n]$	$f[x_0,x_1,x_n]$	$f[x_0,x_1,x_2,x_n]$	…	$\underline{f[x_0,x_1,\cdots,x_n]}$

为了建立 n 次 Newton插值多项式，求各阶均差需进行 $\frac{1}{2}(n^2+n)$次除法和 n^2+n 次减法。得出公式（2.4.8)后，可利用 Horner 算法（1.4.6）求值，每次求值需 n 次乘法、n 次减法和 n 次加法。与 Lagrange 插值相比，Newton插值大大节省了工作量。

2.4.11 例 考虑例（2.3.5）中的问题。

解 利用 Newton 插值，均差表如下：

0.00	$\underline{0.00000}$			
0.20	0.20134	$\underline{1.0067}$		
0.30	0.30452	1.0151	$\underline{0.08367}$	
0.50	0.52110	1.0422	0.11833	$\underline{0.17332}$

Newton 均差插值多项式为

$$N_3(x)=1.0067x+0.08367x(x-0.20)$$
$$+0.17332x(x-0.20)(x-0.30)$$

由此可算出

$$f(0.23)\approx N_3(0.23)\approx 0.23203$$

因 $f(x)=\sinh x$，$f^{(4)}(x)=\sinh x$，余项为

$$R_3(x)=\frac{1}{4!}x(x-0.20)(x-0.30)(x-0.50)\sinh\xi$$

$$(0<\xi<0.5)$$

可得误差估计

$$|R_3(0.23)|<\frac{1}{24}\cdot 0.23\cdot 0.03\cdot 0.07\cdot 0.27\cdot 0.53$$

$$\approx 0.000003$$

如果增加一个节点$x_4=0.60$，则上述均差表增加如下一行：

0.60 | 0.63665 1.0611 0.13596 0.17429 0.00974

插值多项式为

$$N_4(x)=N_3(x)+0.00974x(x-0.20)(x-0.30)(x-0.50)$$

2.5

2.5 差分与等距节点插值

2.5.1 差分

前面的插值公式一般只应用于非等距节点（即节点不是等距离分布）的情形。在实际应用中，常采用等距节点

2.5.1 $x_i=x_0+ih$ $(i=0, \pm 1, \pm 2, \cdots)$

其中 h 是常数，称为**步长**(Step Width)。这时利用差分可以导出计算上更为有效的插值多项式。

用 f_i 表示函数 f 在节点 x_i 的值，即

$f_i=f(x_i)$ $(i=0, \pm 1, \pm 2, \cdots)$

2.5.2 **定义** 令

$$\Delta f_i=f_{i+1}-f_i$$

$$\nabla f_i=f_i-f_{i-1}$$

$$\delta f_i=f_{i+\frac{1}{2}}-f_{i-\frac{1}{2}}$$

这里$f_{i+\frac{1}{2}}=f\left(x_i+\frac{h}{2}\right)$，$f_{i-\frac{1}{2}}=f\left(x_i-\frac{h}{2}\right)$。$\Delta f_i$，$\nabla f_i$，$\delta f_i$ 分别称为 f在 x_i 的**一阶向前差分**(First Forward Difference)，**一阶向后差分**(First Backward Difference)，**一阶中心差分**(First Central Difference)。Δ，∇，δ称为(向前，向后，

中心）差分算子(Difference Operator)。

由此，可递推地定义 n 阶差分为

$$\Delta^n f_i = \Delta^{n-1} f_{i+1} - \Delta^{n-1} f_i$$

$$\nabla^n f_i = \nabla^{n-1} f_i - \nabla^{n-1} f_{i-1}$$

$$\delta^n f_i = \delta^{n-1} f_{i+\frac{1}{2}} - \delta^{n-1} f_{i-\frac{1}{2}}$$

并规定零阶差分为

$$\Delta^0 f_i = \nabla^0 f_i = \delta^0 f_i = f_i$$

2.5.3 例 二阶差分和三阶差分

$$\Delta^2 f_i = \Delta(\Delta f_i) = \Delta f_{i+1} - \Delta f_i = f_{i+2} - 2f_{i+1} + f_i$$

$$\Delta^3 f_i = \Delta^2 f_{i+1} - \Delta^2 f_i = f_{i+3} - 3f_{i+2} + 3f_{i+1} - f_i$$

$$\nabla^2 f_i = \nabla(\nabla f_i) = \nabla f_i - \nabla f_{i-1} = f_i - 2f_{i-1} + f_{i-2}$$

$$\nabla^3 f_i = \nabla^2 f_i - \nabla^2 f_{i-1} = f_i - 3f_{i-1} + 3f_{i-2} - f_{i-3}$$

$$\delta^2 f_i = \delta(\delta f_i) = \delta f_{i+\frac{1}{2}} - \delta f_{i-\frac{1}{2}} = f_{i+1} - 2f_i + f_{i-1}$$

$$\delta^3 f_i = \delta^2 f_{i+\frac{1}{2}} - \delta^2 f_{i-\frac{1}{2}} = f_{i+\frac{3}{2}} - 3f_{i+\frac{1}{2}} + 3f_{i-\frac{1}{2}} - f_{i-\frac{3}{2}}$$

2.5.4 定理 各阶差分可用函数值表示如下：

$$\Delta^n f_i = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f_{n+i-k}$$

$$\nabla^n f_i = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f_{i-k}$$

$$\delta^n f_i = \sum_{k=0}^{n} (-1)^k \binom{n}{k} f_{\frac{n}{2}+i-k}$$

其中$\binom{n}{k} = \dfrac{n(n-1)\cdots(n-k+1)}{k!}$是二项式展开系数。

2.5.5 定理 函数值可用各阶差分表示如下：

$$f_{i+n} = \sum_{k=0}^{n} \binom{n}{k} \Delta^k f_i$$

$$f_{i-n}=\sum_{k=0}^{n}(-1)^k\binom{n}{k}\nabla^k f_i$$

$$f_{i+n}=\sum_{k=0}^{n}\binom{n}{k}\delta^k f_{i+\frac{k}{2}}$$

2.5.6 定理 差分与均差的关系如下：

$\Delta^m f_i=m!h^m f[x_i,x_{i+1},\cdots,x_{i+m}]$

$\nabla^m f_i=m!h^m f[x_{i-m},x_{i-m+1},\cdots,x_i]$

$\delta^{2m+1} f_{i+\frac{1}{2}}=(2m+1)!h^{2m+1} f[x_{i-m},x_{i-m+1},\cdots,x_{i+m+1}]$

$\delta^{2m} f_i=(2m)!h^{2m} f[x_{i-m},x_{i-m+1},\cdots,x_{i+m}]$

以上三个定理均可用数学归纳法证明．由（2.4.6）及定理（2.5.6）可得差分和导数的关系，例如向前差分与导数的关系为

2.5.7 $\Delta^n f_i=h^n f^{(n)}(\xi)$，$\xi\in[x_i,x_{i+n}]$

利用函数值 $f_i(i=0,\pm1,\pm2,\cdots)$计算各阶差分特别简单，仅包含减法运算，可以构造差分表（2.5.8）．

2.5.8 表

f_i	$\Delta(\nabla,\delta)$	$\Delta^2(\nabla^2,\delta^2)$	$\Delta^3(\nabla^3,\delta^3)$
$\vdots$			
f_{-1}	$\vdots$	$\vdots$	$\vdots$
	$\Delta f_{-1}(\nabla f_0,\delta f_{-1/2})$		
f_0		$\Delta^2 f_{-1}(\nabla^2 f_1,\delta^2 f_0)$	
	$\Delta f_0(\nabla f_1,\delta f_{-1/2})$		$\Delta^3 f_{-1}(\nabla^3 f_2,\delta^3 f_{1/2})$
f_1		$\Delta^2 f_0(\nabla^2 f_2,\delta^2 f_1)$	
	$\Delta f_1(\nabla f_2,\delta f_{3/2})$	$\vdots$	$\vdots$
f_2			
$\vdots$	$\vdots$		

2.5.2 Newton差分插值公式

利用定理（2.5.6），在 Newton 均差插值公式(2.4.8)中将均差替换为差分，可以得出不同形式的插值公式.

在以 $x_i, x_{i+1}, \cdots, x_{i+n}$ 为节点的 Newton 均差插值公式中，用向前差分代替均差，并令 $x=x_i+th, 0\leqslant t\leqslant n$，可得

2.5.9 $$N_n(x_i+th)=f_i+\binom{t}{1}\Delta f_i+\binom{t}{2}\Delta^2 f_i+\cdots+\binom{t}{n}\Delta^n f_i$$

此即 **Newton向前差分公式**，式中记号

$$\binom{t}{k}=\frac{t(t-1)\cdots(t-k+1)}{k!}$$

t 是变量. 这时

$$W_{n+1}(x)=\prod_{k=0}^{n}(x-x_{i+k})=t(t-1)\cdots(t-n)h^{n+1}$$

根据（2.2.12），余项可以写成

2.5.10 $$R_n(x_i+th)=\binom{t}{n+1}h^{n+1}f^{(n+1)}(\xi),\quad \xi\in[x_i,\ x_{i+n}]$$

类似地，在以 $x_i, x_{i-1}, \cdots, x_{i-n}$（从大到小排列）为节点的 Newton 均差插值公式中，用向后差分代替均差，并令 $x=x_i+th,\ -n\leqslant t\leqslant 0$，可得

2.5.11 $N_n(x_i+th)$

$$=f_i+\binom{t}{1}\nabla f_i+\binom{t+1}{2}\nabla^2 f_i+\cdots+\binom{t+n-1}{n}\nabla^n f_i$$

称为 Newton **向后差分公式**. 余项可写成

2.5.12 $$R_n(x_i+th)=\binom{t+n}{n+1}h^{n+1}f^{(n+1)}(\xi),\quad \xi\in[x_{i-n}, x_i]$$

2.5.3 Gauss公式

在 $n+1$ 个节点按 $x_i, x_{i+1}, x_{i-1}, x_{i+2}, x_{i-2}, \cdots$ 顺序排列的 Newton均差插值公式中，根据定理(2.5.6)，用中心差商代替

均差，并令 $x=x_l+th$，可得又一种插值公式。当 $n=2m$时，节点为 $x_l, x_{l+1}, x_{l-1}, \cdots, x_{l+m}, x_{l-m}(-m\leqslant t\leqslant m)$

2.5.13 $$G_n(t)=N_n(x_l+th)=f_l+\binom{t}{1}\delta f_{l+\frac{1}{2}}+\binom{t}{2}\delta^2 f_l$$
$$+\binom{t+1}{3}\delta^3 f_{l+\frac{1}{2}}+\binom{t+1}{4}\delta^4 f_l+\cdots$$
$$+\binom{t+m-1}{2m-1}\delta^{2m-1}f_{l+\frac{1}{2}}+\binom{t+m-1}{2m}\delta^{2m}f_l$$

余项可写成

2.5.14 $$R_n(x_l+th)=\binom{t+m}{2m+1}h^{2m+1}f^{(2m+1)}(\xi)$$
$$\xi\in[x_{l-m},\ x_{l+m}]$$

当 $n=2m+1$时，增加一个节点 x_{l+m+1}，$-m\leqslant t\leqslant m+1$，只须在（2.5.15）中加一项，

2.5.15 $$G_n(t)=f_l+\binom{t}{1}\delta f_{l+\frac{1}{2}}+\binom{t}{2}\delta^2 f_l+\binom{t+1}{3}\delta^3 f_{l+\frac{1}{2}}$$
$$+\binom{t+1}{4}\delta^4 f_l+\cdots+\binom{t+m-1}{2m-1}\delta^{2m-1}f_{l+\frac{1}{2}}$$
$$+\binom{t+m-1}{2m}\delta^{2m}f_l+\binom{t+m}{2m+1}\delta^{2m+1}f_{l+\frac{1}{2}}$$

余项变为

2.5.16 $$R_n(x_l+th)=\binom{t+m}{2m+2}h^{2m+2}f^{(2m+2)}(\xi)$$
$$\xi\in[x_{l-m},\ x_{l+m+1}]$$

（2.5.15）与（2.5.15）称为 Gauss 向前公式。

类似地，如果 $n+1$ 个节点改而按 x_l，x_{l-1}，x_{l+1}，x_{l-2}，x_{l+2}，…顺序排列，$x=x_l+th$，则当 $n=2m$ 时，$-m\leqslant t\leqslant m$，有

2.5.17 $$G_n(t)=f_i+\binom{t}{1}\delta f_{i-\frac{1}{2}}+\binom{t+1}{2}\delta^2 f_i+\binom{t+1}{3}\delta^3 f_{i-\frac{1}{2}}$$

$$+\binom{t+2}{4}\delta^4 f_i+\cdots+\binom{t+m-1}{2m-1}\delta^{2m-1} f_{i-\frac{1}{2}}+\binom{t+m}{2m}\delta^{2m} f_i$$

余项为

2.5.18 $$R_n(x_i+th)=\binom{t+m}{2m+1}h^{2m+1}f^{(2m+1)}(\xi)$$

$$\xi\in[x_{i-m},\ x_{i+m}]$$

当 $n=2m+1$ 时，$-m-1\leqslant t\leqslant m$，有

2.5.19 $$G_n(t)=f_i+\binom{t}{1}\delta f_{i-\frac{1}{2}}+\binom{t+1}{2}\delta^2 f_i+\binom{t+1}{3}\delta^3 f_{i-\frac{1}{2}}$$

$$+\binom{t+2}{4}\delta^4 f_i+\cdots+\binom{t+m-1}{2m-1}\delta^{2m-1} f_{i-\frac{1}{2}}$$

$$+\binom{t+m}{2m}\delta^{2m} f_i+\binom{t+m}{2m+1}\delta^{2m+1} f_{i-\frac{1}{2}}$$

余项为

2.5.20 $$R_n(x_i+th)=\binom{t+m+1}{2m+2}h^{2m+2}f^{(2m+2)}(\xi)$$

$$\xi\in[x_{i-m-1},\ x_{i+m}]$$

(2.5.17) 与 (2.5.19) 称为 Gauss 向后公式。

2.5.4 Stirling公式

取 Gauss 向前公式(2.5.13)与 Gauss 向后公式(2.5.17)的平均，即可得到以 x_{i-m}，…，x_i，…，x_{i+m} 为节点的插值公式

2.5.21 $$S_n(t)=f_i+\frac{1}{2}\binom{t}{1}(\delta f_{i+\frac{1}{2}}+\delta f_{i-\frac{1}{2}})$$

$$+\frac{1}{2}\left[\binom{t}{2}+\binom{t+1}{2}\right]\delta^2 f_i+\frac{1}{2}\binom{t+1}{3}(\delta^3 f_{i+\frac{1}{2}}+\delta^3 f_{i-\frac{1}{2}})$$

$$+\cdots+\frac{1}{2}\left[\binom{t+m-1}{2m}+\binom{t+m}{2m}\right]\delta^{2m} f_i$$

这里 $-m\leqslant t=\frac{x-x_i}{h}\leqslant m$，$n=2m$. 余项为

2.5.22 $$R_n(x_i+th)=\binom{t+m}{2m+1}h^{2m+1}f^{(2m+1)}(\xi)$$

$$\xi\in[x_{i-m},\ x_{i+m}]$$

(2.5.21) 称为 Stirling (斯特林) 公式. 这个公式用的节点个数是奇数.

2.5.5 Bessel 公式

在 Gauss 向后公式 (2.5.19) 中用 $i+1$ 代替 i，同时用 $t-1$ 代替 t，然后与 Gauss 向前公式 (2.5.15) 取平均，可得到以 $x_{i-m},\cdots,x_i,\cdots,x_{i+m},x_{i+m+1}$ 为节点的又一个插值公式

2.5.23 $$B_n(t)=\frac{1}{2}(f_i+f_{i+1})+\frac{1}{2}\left[\binom{t}{1}+\binom{t-1}{1}\right]\delta f_{i+\frac{1}{2}}$$

$$+\frac{1}{2}\binom{t}{2}(\delta^2 f_i+\delta^2 f_{i+1})+\frac{1}{2}\left[\binom{t+1}{3}+\binom{t}{3}\right]\delta^3 f_{i+\frac{1}{2}}$$

$$+\cdots+\frac{1}{2}\left[\binom{t+m}{2m+1}+\binom{t+m-1}{2m+1}\right]\delta^{2m+1} f_{i+\frac{1}{2}}$$

这里 $-m\leqslant t=\frac{x-x_i}{h}\leqslant m+1$，$n=2m+1$. 余项为

2.5.24 $$R_n(x_i+th)=\binom{t+m}{2m+2}h^{2m+2}f^{(2m+2)}(\xi)$$

$$\xi\in[x_{i-m},x_{i+m+1}]$$

(2.5.23) 称为 Bessel（贝塞耳）公式．这个公式只能用于偶数个节点．

2.5.6 Everett 公式

在 Gauss 向前公式（2.5.15）中，如果利用关系式

$\delta f_{i+\frac{1}{2}}=f_{i+1}-f_i,\quad \delta^3 f_{i+\frac{1}{2}}=\delta^2 f_{i+1}-\delta^2 f_i,\quad\cdots$

将奇数阶中心差分都变换为偶数阶中心差分，则可得到以 $x_{i-m},\cdots,x_i,\cdots,x_{i+m},x_{i+m+1}$ 为节点的插值公式

2.5.25
$$E_n(t)=\binom{1-t}{1}f_i+\binom{(1-t)+1}{3}\delta^2 f_i+\binom{(1-t)+2}{5}\delta^4 f_i+\cdots+\binom{(1-t)+m}{2m+1}\delta^{2m}f_i+\binom{t}{1}f_{i+1}+\binom{t+1}{3}\delta^2 f_{i+1}+\binom{t+2}{5}\delta^4 f_{i+1}+\cdots+\binom{t+m}{2m+1}\delta^{2m}f_{i+1}$$

这里 $-m\leqslant t=\dfrac{x-x_i}{h}\leqslant m+1,\quad n=2m+1$．余项为

2.5.26
$$R_n(x_i+th)=\binom{t+m}{2m+2}h^{2m+2}f^{(2m+2)}(\xi)\qquad \xi\in[x_{i-m},x_{i+m+1}]$$

(2.5.25) 称为 Everett（埃弗瑞特）公式．这个公式利用了偶数个节点．

2.5.7 Steffensen 公式

在 Gauss 向前公式（2.5.13）中，如果利用关系式

$\delta^2 f_i=\delta f_{i+\frac{1}{2}}-\delta f_{i-\frac{1}{2}},\quad \delta^4 f_i=\delta^3 f_{i+\frac{1}{2}}-\delta^3 f_{i-\frac{1}{2}},\quad\cdots$

将偶数阶中心差分变换为奇数阶中心差分，则可得到以 $x_{i-m},\cdots,x_i,\cdots,x_{i+m}$ 为节点的插值公式

2.5.27
$$S_n(t)=f_i+\binom{t+1}{2}\delta f_{i+\frac{1}{2}}+\binom{t+2}{4}\delta^3 f_{i+\frac{1}{2}}$$

$$+\cdots+\binom{t+m}{2m}\delta^{2m-1}f_{l+\frac{1}{2}}-\binom{-t+1}{2}\delta f_{l-\frac{1}{2}}$$

$$-\binom{-t+2}{4}\delta^{3}f_{l-\frac{1}{2}}-\cdots-\binom{-t+m}{2m}\delta^{2m-1}f_{l-\frac{1}{2}}$$

这里 $-m\leqslant t=\frac{x-x_l}{h}\leqslant m$，$n=2m$。余项为

2.5.28 $$R_n(x_l+th)=\binom{t+m}{2m+1}h^{2m+1}f^{(2m+1)}(\xi)$$

$$\xi\in[x_{l-m},x_{l+m}]$$

（2.5.27）称为 Steffensen（斯梯芬森）公式。这个公式利用了奇数个节点。在这个公式中采用$\binom{-t+k}{2k}$的写法主要是为了形式上的对称，实际上，

$$\binom{-t+k}{2k}=\binom{t+k-1}{2k}$$

2.6

2.6 插值公式的几个问题

2.6.1 插值公式的使用

尽管最一般插值公式——Lagrange 公式——计算工作量较大，但它仍有重要的应用。例如，数值积分和微分的许多特殊的公式与方法就是从 Lagrange 插值公式导出的。对于非等距节点的情形来说，利用 Aitken 法和 Newton均差插值公式则可通过逐步增加节点，产生一个逼近于被插函数的多项式序列。它们适合于在计算机上计算，编制程序比较简单，而且可以自动选节点并逐步比较精度。

在等距节点的情形下，利用差分建立的各种插值公式可以节省大量的计算工作量。计算 1 至 n 阶差分总共只需进行 $n(n+1)/2$次减法。概括地说，它们都是用来产生和分析逼近于被插函数的多项式序列。Newton 向前差分与向后差分公式分别适用于新节点总是等距地沿增大的方向引入和总是等距地沿变小的方向引入，或者分别适用于函数的离散数据

表的表首附近的计算和表末附近的计算．Gauss，Stirling和Bessel等中心差分的公式适用于在一节点两边，新的节点等距交替地引入，或者适用于在离散数据表的中间附近的计算．

以 $f(x)=\sin x$ 为例，说明等距节点各种插值公式的应用．下面是经计算得到的一个差分表：

x	$f(x)$	$\Delta(\nabla,\delta)$	$\Delta^2(\nabla^2,\delta^2)$	$\Delta^3(\nabla^3,\delta^3)$	$\Delta^4(\nabla^4,\delta^4)$	$\Delta^5(\nabla^5,\delta^5)$
1.0	0.84147					
		0.04974				
1.1	0.89121		−0.00891			
		0.04083		−0.00040		
1.2	0.93204		−0.00931		0.00008	
		0.03152		−0.00032		0.00002
1.3	0.96356		−0.00963		0.00010	
		0.02189		−0.00022		0.00001
1.4	0.98545		−0.00985		0.00011	
		0.01204		−0.00011		−0.00003
1.5	0.99749		−0.00996		0.00008	
		0.00208		−0.00003		0.00004
1.6	0.99957		−0.00999		0.00012	
		−0.00791		0.00009		
1.7	0.99166		−0.00990			
		−0.01781				
1.8	0.97385					

检验差分表的正确性可根据每列差分之和应等于前一列的最后一数与最前一数之差．上述表中五阶差分已近于零，因此只能用至四阶差分．

为了求 $f(1.02)$ 的近似值，使用 Newton 向前差分公式

$$N_4(1+0.1t)=0.84147+0.04974t-0.00891\frac{t(t-1)}{2!}$$

$$-0.00040\frac{t(t-1)(t-2)}{3!}+0.00008\frac{t(t-1)(t-2)(t-3)}{4!}$$

取 $t=0.2$，得

$f(1.02)\approx N_4(1.02)\approx 0.85211$

为了求 $f(1.75)$ 的近似值，使用 Newton 向后差分公式

$$N_4(1.8+0.1t)=0.97385-0.01781t-0.00990\frac{t(t+1)}{2!}$$

$$+0.00009\frac{t(t+1)(t+2)}{3!}$$

$$+0.00012\frac{t(t+1)(t+2)(t+3)}{4!}$$

取 $t=-0.5$，得

$f(1.75)\approx N_4(1.75)\approx 0.98398$

为了求 $f(1.22)$ 的近似值，使用中心差分插值公式。Gauss 公式总可用 Stirling 公式或 Bessel 公式等价地代替。Stirling 公式

$$S_4(t)=0.93204+\frac{1}{2}(0.03152+0.04083)t-0.00931\frac{t^2}{2!}$$

$$-\frac{1}{2}(0.00032+0.00040)\frac{t(t^2-1)}{3!}+0.00008\frac{t^2(t^2-1)}{4!}$$

其中 $t=\frac{x-1.2}{0.1}$。当 $x=1.22$ 时 $t=0.2$，得

$f(1.22)\approx S_4(0.2)\approx 0.93910$

Bessel 公式是

$$B_4(t)=\frac{1}{2}(0.93204+0.96356)+0.03152\left(t-\frac{1}{2}\right)$$

$$+\frac{1}{2}(-0.00963-0.00931)\frac{t(t-1)}{2!}-0.00032\frac{t(t-1)\left(t-\frac{1}{2}\right)}{3!}$$

$$+\frac{1}{2}(0.00008+0.00010)\frac{t(t^2-1)(t-2)}{4!}$$

同样有

$f(1.22)\approx B_4(0.2)\approx 0.93910$

再使用 Everett 公式

$$E_5(t)=0.93204(1-t)+0.00931\frac{t(t-1)(t-2)}{3!}$$

$$-0.00008\frac{(t+1)t(t-1)(t-2)(t-3)}{5!}$$

$$+0.96356t-0.00963\frac{(t+1)t(t-1)}{3!}$$

$$+0.00010\frac{(t+2)(t+1)t(t-1)(t-2)}{5!}$$

也可得到

$f(1.22)\approx E_5(0.2)\approx 0.93910$

最后，使用 Steffensen 公式

$$S_4(t)=0.93204+0.03152\frac{(t+1)t}{2!}$$

$$-0.00032\frac{(t+2)(t+1)t(t-1)}{4!}-0.04083\frac{t(t-1)}{2!}$$

$$+0.00040\frac{(t+1)t(t-1)(t-2)}{4!}$$

仍然有

$f(1.22)\approx S_4(t)\approx 0.93910$

2.6.2 Fraser 图表

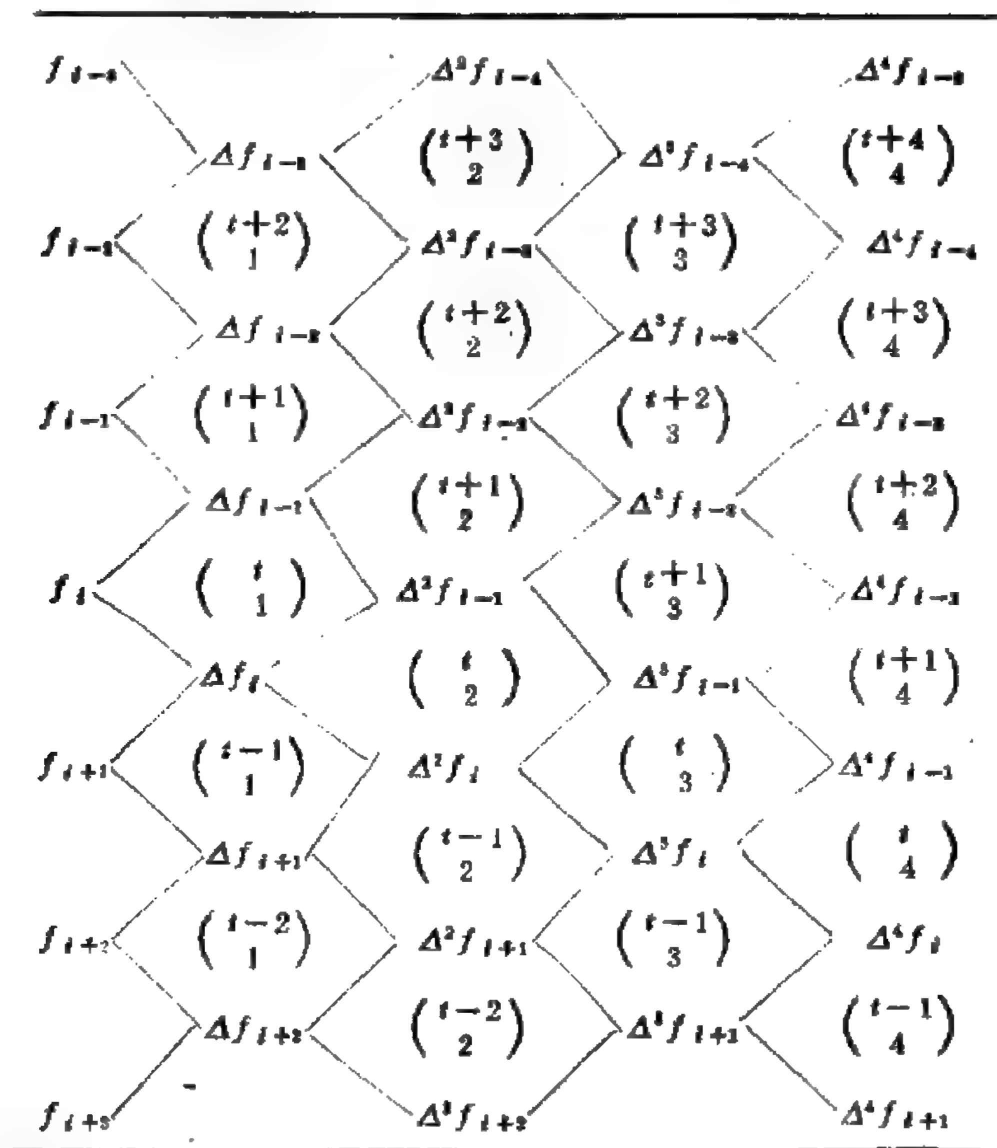
图表（2.6.1）是在差分表中加上所示的连结线与二项式系数而构成的，称为 Fraser（弗雷瑟）图表。它提供了建立插值多项式的一个一般的方法。

在（2.6.1）的表中，从第一列的一个函数值出发，可以沿着由线段连结而成的多种多样的路径前进。当到达一差分时，允许沿着该差分周围的四条线段的任一条继续前进；当到达一函数值时，允许沿该函数值开头的两条线段中的任一条继续前进。每条路径最后在一个差分上结束。

利用表（2.6.1）建立插值公式的过程是：任意选定一条路径，写下路径出发时的函数值作为公式的第一项，然后在沿路径前进的过程中对每一线段给公式附加一个项。对每

一条从左到右且有正（负）斜率的线段，附加的项是线段终止点的差分与该差分下面（上面）二项式系数的乘积；对每一条从右到左的线段，附加的项是该线段从左到右时的附加的项乘－1．

2.6.1 Fraser 图表



表中

2.6.2 $t=\dfrac{x-x_i}{h}$，$\dbinom{t}{k}=\dfrac{t(t-1)\cdots(t-k+1)}{k!}$

插值多项式最常用的差分公式可以由 Fraser 图表法得出．例如，从 f_i 出发沿负斜率线段向右前进，路径经Δf_i，

$\Delta^2 f_i$，…，直至 $\Delta^n f_i$ 结束，即得 Newton 向前差分公式（2.5.9）。注意到关系式

2.6.3 $\Delta^n f_i = \nabla^n f_{i+n} = \delta^n f_{i+\frac{n}{2}}$

Fraser 图表中可用向后差分或中心差分替换向前差分。这样，不难看出，从 f_i 出发沿正斜率线段向右前进，路径经 Δf_{i-1}，$\Delta^2 f_{i-2}$，…，直至 $\Delta^n f_{i-n}$ 结束，即得 Newton 向后差分公式（2.5.11）；从 f_i 出发沿斜率先负后正且正负相间的线段向右前进，锯齿形的路径经 Δf_i，$\Delta^2 f_{i-1}$，$\Delta^3 f_{i-1}$，$\Delta^4 f_{i-2}$，…，可得 Gauss 向前公式（2.5.13）与（2.5.15）；类似地，从 f_i 出发沿斜率先正后负的线段向右前进，锯齿形路径经 Δf_{i-1}，$\Delta^2 f_{i-1}$，$\Delta^3 f_{i-2}$，$\Delta^4 f_{i-2}$，…，则可得 Gauss 向后公式（2.5.17）与（2.5.19）。

可以证明，由 Fraser 图表得到的公式等价于对结束的差分所涉及的节点进行插值的插值多项式。因此，由Fraser 图表可构造多种多样的插值公式。

2.6.3 插值公式求值用表

为了便于插值公式的求值，一些等距节点插值公式有简明的用表。

2.6.4 Newton插值 （2.5.9）与（2.5.11）改写成（$n=5$）

$$N_5(x_i + th) = f_i + t\Delta f_i + C_2(t)\Delta^2 f_i + C_3(t)\Delta^3 f_i + C_4(t)\Delta^4 f_i + C_5(t)\Delta^5 f_i$$

$$N_5(x_i - th) = f_i - t\nabla f_i + C_2(t)\nabla^2 f_i - C_3(t)\nabla^3 f_i + C_4(t)\nabla^4 f_i - C_5(t)\nabla^5 f_i$$

t	$C_2(t)$	$C_3(t)$	$C_4(t)$	$C_5(t)$
−1.0	1.00000	−1.00000	1.00000	−1.00000
−0.9	0.85500	−0.82650	0.80584	−0.78972
−0.8	0.72000	−0.67200	0.63840	−0.61286
−0.7	0.59500	−0.53550	0.49534	−0.46562
−0.6	0.48000	−0.41600	0.37440	−0.34445
−0.5	0.37500	−0.31250	0.27344	−0.24609
−0.4	0.28000	−0.22400	0.18040	−0.16755
−0.3	0.19500	−0.14950	0.12334	−0.10607
−0.2	0.12000	−0.08800	0.07040	−0.05914
−0.1	0.05500	−0.03850	0.02984	−0.02447
0.0	0.00000	0.00000	0.00000	0.00000
0.1	−0.04500	0.02850	−0.02066	0.01612
0.2	−0.08000	0.04800	−0.03360	0.02554
0.3	−0.10500	0.05950	−0.04016	0.02972
0.4	−0.12000	0.06400	−0.04160	0.02995
0.5	−0.12500	0.06250	−0.03906	0.02734
0.6	−0.12000	0.05600	−0.03360	0.02285
0.7	−0.10500	0.04550	−0.02616	0.01727
0.8	−0.08000	0.03200	−0.01760	0.01126
0.9	−0.04500	0.01650	−0.00866	0.00537
1.0	0.00000	0.00000	0.00000	0.00000

插值时用 $t>0$。

2.6.5 Stirling插值 (2.5.21) 写成($n=4$)

$$S_4(t)=f_i+\frac{1}{2}t\,(\delta f_{i+\frac{1}{2}}+\delta f_{i-\frac{1}{2}})+C_2(t)\delta^2 f_i$$

$$+\frac{1}{2}C_3(t)(\delta^3 f_{i+\frac{1}{2}}+\delta^3 f_{i-\frac{1}{2}})+C_4(t)\delta^4 f_i$$

$$t=\frac{x-x_i}{h}$$

t	$C_2(t)$	$C_3(t)$	$C_4(t)$	t
0.0	0.00000	0.00000	0.00000	0.0
0.1	0.00500	−0.01650*	−0.00041	−0.1
0.2	0.02000	−0.03200*	−0.00160	−0.2
0.3	0.04500	−0.04550*	−0.00341	−0.3
0.4	0.08000	−0.05600*	−0.00560	−0.4
0.5	0.12500	−0.06250*	−0.00781	−0.5
0.6	0.18000	−0.06400*	−0.00960	−0.6
0.7	0.24500	−0.05950*	−0.01041	−0.7
0.8	0.32000	−0.04800*	−0.00960	−0.8
0.9	0.40500	−0.02850*	−0.00641	−0.9
1.0	0.50000	0.00000	0.00000	−1.0

* 从右边列读t时反号

2.6.6 Bessel 插值（2.5.23）写成（$n=5$）

$$B_5(t)=\frac{1}{2}(f_i+f_{i+1})+(t-\frac{1}{2})\delta f_{i+\frac{1}{2}}$$

$$+\frac{1}{2}C_2(t)(\delta^2 f_i+\delta^2 f_{i+1})+C_3(t)\,\delta^3 f_{i+\frac{1}{2}}$$

$$+\frac{1}{2}C_4(t)(\delta^4 f_i+\delta^4 f_{i+1})+C_5(t)\delta^5 f_{i+\frac{1}{2}},\quad t=\frac{x-x_i}{h}$$

t	$C_2(t)$	$C_3(t)$	$C_4(t)$	$C_5(t)$	t
0.0	0.00000	0.00000	0.00000	0.00000	1.0
0.1	−0.04500	0.00600*	0.00784	−0.00063*	0.9
0.2	−0.08000	0.00800*	0.01440	−0.00086*	0.8
0.3	−0.10500	0.00700*	0.01934	−0.00077*	0.7
0.4	−0.12000	0.00400*	0.02240	−0.00045*	0.6
0.5	−0.12500	0.00000	0.02344	0.00000	0.5

* 从右边列读t时反号

2.6.7 Everett 插值（2.5.25）写成（$n=5$）

$$E_5(t)=(1-t)f_i+C_2(t)\delta^2 f_i+C_4(t)\delta^4 f_i+tf_{i+1}$$
$$+C_2(1-t)\delta^2 f_{i+1}+C_4(1-t)\delta^4 f_{i+1},\quad t=\frac{x-x_i}{h}$$

t	$C_2(t)$	$C_4(t)$
0.0	0.00000	0.00000
0.1	−0.02850	0.00455
0.2	−0.04800	0.00806
0.3	−0.05950	0.01044
0.4	−0.06400	0.01165
0.5	−0.06250	0.01172
0.6	−0.05600	0.01075
0.7	−0.04550	0.00890
0.8	−0.03200	0.00634
0.9	−0.01650	0.00329
1.0	0.00000	0.00000

2.6.8 Steffensen插值 (2.5.27)写成 ($n=4$)

t	$C_1(t)$	$C_3(t)$
−0.5	−0.12500	0.02344
−0.4	−0.12000	0.02240
−0.3	−0.10500	0.01934
−0.2	−0.08000	0.01440
−0.1	−0.04500	0.00784
0.0	0.00000	0.00000
0.1	0.05500	−0.00806
0.2	0.12000	−0.01760
0.3	0.19500	−0.02616
0.4	0.28000	−0.03360
0.5	0.37500	−0.03906

$$S_4(t)=f_i+C_1(t)\delta f_{i+\frac{1}{2}}+C_3(t)\delta^3 f_{i+\frac{1}{2}}$$

$$-C_1(-t)\delta f_{i-\frac{1}{2}}-C_3(-t)\delta^3 f_{i-\frac{1}{2}},\quad t=\frac{x-x_i}{h}$$

当函数f的差分表列出之后，根据以上各表及相应的插值公式，可以方便地求$f(x)=f(x_i+th)$的近似值。

2.7

2.7 Hermite插值

2.7.1 一般提法

在应用中，有时需用Hermite插值，它是Lagrange插值的推广，除了考虑函数值外还要考虑到导数值。Hermite插值的一般提法是：设函数f在节点x_0，$x_1,\cdots,x_n$的函数值与导数值为

$$f(x_i)=f_i,\ f'(x_i)=f_i',\cdots,f^{(\alpha_i-1)}(x_i)=f_i^{(\alpha_i-1)}$$
$$(i=0,1,\cdots,n)$$

其中$\alpha_0,\alpha_1,\cdots,\alpha_n$是正整数。寻求一个次数尽可能低的多项式$H$，使得满足条件

2.7.1 $$H^{(k)}(x_i)=f_i^{(k)}\quad(k=0,1,\cdots,\alpha_i-1,\quad i=0,1,\cdots,n)$$

可以证明，存在唯一的满足插值条件（2.7.1）的次数不超过$\alpha=\sum\limits_{i=0}^{n}\alpha_i-1$的多项式$H$，即所谓Hermite插值多项式，其形式可写成

2.7.2 $$H(x)=\sum_{i=0}^{n}\sum_{k=0}^{\alpha_i-1}f_i^{(k)}h_{ik}(x)$$

其中h_{ki}是α次多项式，满足条件

$$h_{ik}^{(l)}(x_j)=\begin{cases}1 & (j=i\ \text{且}\ l=k)\\ 0 & (\text{其它})\end{cases}$$

作为一种具体的情形，设

2.7.3 $f(x_i)=f_i,\ f'(x_i)=f_i' \quad (i=0,1,\cdots,n)$

要求构造一个次数不超过 $2n+1$ 的多项式，记作 H_{2n+1}，使得

2.7.4 $H_{2n+1}(x_i)=f_i,\quad H'_{2n+1}(x_i)=f_i' \quad (i=0,1,\cdots,n)$

从几何上看，即要求 H_{2n+1} 与 f 在 $n+1$ 个节点处相切，这时 $x_0,x_1,\cdots,x_n$ 称为**二重节点**。

2.7.2 插值多项式的建立

先考虑特殊情况：求 $2n+1$ 次的多项式 α_i 和 $\beta_i(0\leqslant i\leqslant n)$，满足条件

2.7.5 $\alpha_i(x_j)=\delta_{ij},\quad \alpha_i'(x_j)=0 \quad (j=0,1,\cdots,n)$

2.7.6 $\beta_i(x_j)=0,\quad \beta_i'(x_j)=\delta_{ij} \quad (j=0,1,\cdots,n)$

这里 δ_{ij} 是 Kronecker 记号，其定义见（2.2.1）。这是两个最简单的 Hermite 插值问题.

由条件（2.7.5）与（2.7.6），及多项式的根和重根的性质，可直接确定 β_i，

2.7.7 $\beta_i(x)=(x-x_i)l_i^2(x)$

并且可确定 α_i，

$\alpha_i(x)=[a(x-x_i)+1]l_i^2(x)$

其中 a 是待定常数。以上两式中的 l_i 是由（2.2.3）定义的 Lagrange 插值的基函数。根据条件 $\alpha_i'(x_i)=0$ 可定出 a，并由此得出

2.7.8 $$\alpha_i(x)=\left[1-2(x-x_i)\sum_{\substack{j=0\\ j\neq i}}^{n}\frac{1}{x_i-x_j}\right]l_i^2(x)$$

α_i 与 $\beta_i(i=0,1,\cdots,n)$ 是关于插值条件（2.7.4）的插值问题的基函数，Hermite 插值多项式 H_{2n+1} 明显是这些基函数的线性组合：

2.7.9 $H_{2n+1}(x)=\sum_{i=0}^{n} f_i\alpha_i(x)+\sum_{i=0}^{n} f_i'\beta_i(x)$

2.7.3 余项

设函数 f 在区间$[a,b]$上存在 $2n+2$阶导数，$x_0,x_1,\cdots,x_n\in[a,b]$，且用多项式 H_{2n+1} 近似 f，R_{2n+1} 是插值余项。可以证明

2.7.10 $R_{2n+1}(x)=f(x)-H_{2n+1}(x)=\dfrac{f^{(2n+2)}(\xi)}{(2n+2)!}W_{n+1}^2(x)$

其中$\xi\in[a,b]$且依赖于x，W_{n+1}仍由（2.2.7）定义。

当$n=1$，节点为x_0，x_1时，基函数为

$$\alpha_0(x)=\left(1+2\frac{x-x_0}{x_1-x_0}\right)\left(\frac{x-x_1}{x_0-x_1}\right)^2$$

$$\alpha_1(x)=\left(1+2\frac{x-x_1}{x_0-x_1}\right)\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

$$\beta_0(x)=(x-x_0)\left(\frac{x-x_1}{x_0-x_1}\right)^2$$

$$\beta_1(x)=(x-x_1)\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

插值多项式为H_3

2.7.11 $$H_3(x)=\left[f_0\cdot\left(1+2\frac{x-x_0}{x_1-x_0}\right)+f_0'\cdot(x-x_0)\right]\left(\frac{x-x_1}{x_0-x_1}\right)^2+\left[f_1\cdot\left(1+2\frac{x-x_1}{x_0-x_1}\right)+f_1'\cdot(x-x_1)\right]\left(\frac{x-x_0}{x_1-x_0}\right)^2$$

余项为R_3

2.7.12 $R_3(x)=\dfrac{f^{(4)}(\xi)}{4!}(x-x_0)^2(x-x_1)^2$

其中 ξ 在 x_0 与 x_1 之间。

2.7.13 例 求多项式 p，满足条件

$p(x_j)=f(x_j)\quad(j=0,1,2)$

$p'(x_1)=f'(x_1)$

并求余项表达式。

解 为了保证满足 $p(x_j)=f(x_j)$，$j=0,1,2$，设

$$p(x)=f(x_0)+f[x_0,x_1](x-x_0)$$
$$+f[x_0,x_1,x_2](x-x_0)(x-x_1)+A(x-x_0)(x-x_1)(x-x_2)$$

这个等式右端前三项是以 x_0,x_1,x_2 为节点的 Newton 均差插值公式，A 是待定常数。利用条件 $p'(x_1)=f'(x_1)$ 定出常数

$$A=\frac{f'(x_1)-f[x_0,x_1]-(x_1-x_0)f[x_0,x_1,x_2]}{(x_1-x_0)\ (x_1-x_2)}$$

因余项 $R(x)=f(x)-p(x)$ 满足条件

$R(x_j)=0\quad(j=0,1,2)$

$R'(x_1)=0$

可设

$R(x)=K(x)(x-x_0)(x-x_1)^2(x-x_2)$

其中 K 是待定函数。应用微分学中著名的 Rolle（罗尔）定理可定出K，最后有

$$R(x)=\frac{1}{4!}f^{(4)}(\xi)(x-x_0)(x-x_1)^2(x-x_2)$$

其中 ξ 在 x_0,x_1,x_2,x 之间。

2.7.14 例 求多项式 p，满足条件

$p(x_j)=f(x_j)\quad(j=0,1)$，$p^{(k)}(x_0)=f^{(k)}(x_0)\quad(k=1,2)$

并求余项表达式。

解 为了满足条件 $p(x_j)=f(x_j)$，$j=0,1$，可设

$$p(x)=f(x_0)+f[x_0,x_1](x-x_0)+(Ax+B)(x-x_0)(x-x_1)$$

其中 A与B 是待定常数，可由条件$p^{(k)}(x_0)=f^{(k)}(x_0)\quad(k=0,1)$ 来确定。经计算得到

$$A=\frac{f''(x_0)}{2(x_0-x_1)}+\frac{f[x_0,x_1]-f'(x_0)}{(x_0-x_1)^2}$$

$$B=\frac{f'(x_0)-f[x_0,x_1]}{x_0-x_1}-Ax_0$$

余项为

$$R(x)=f(x)-p(x)=\frac{1}{4!}f^{(4)}(\xi)(x-x_0)^3(x-x_1)$$

其中 ξ 在 x_0,x_1,x 之间。

2.8

2.8 分段低次插值

2.8.1 高次插值的问题

利用插值多项式逼近函数时，并不是插值多项式的次数越高逼近的效果必定越好。实际情况是，当插值节点无限增多时，次数趋于无穷的插值多项式序列并不一定收敛于被插函数。一个有名的实例是定义在区间 $[-5,5]$ 上的函数 f,

$$f(x)=\frac{1}{1+x^2}$$

它是由 Runge（龙格）提供的。对每一个 $n(n=1,2,\cdots)$，由等距节点集

$$S_n=\left\{x_i=-5+10\frac{i}{n}\,\middle|\, i=0,1,\cdots,n\right\}$$

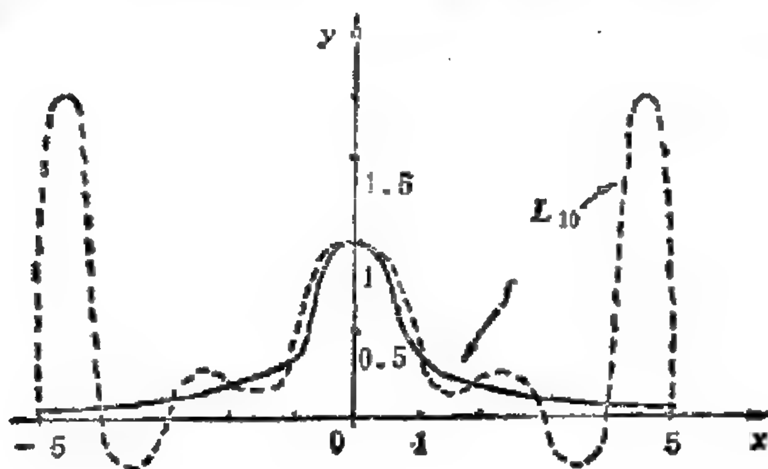
决定一个 Lagrange 插值多项式 L_n,

$$L_n(x)=\sum_{i=1}^{n}\frac{1}{1+x_i^2}l_i(x)$$

插值多项式序列 $\{L_n\}_{n=1}^{\infty}$ 在区间 $[-5,5]$ 上并不收敛于 f。图(2.8.1)中给出了曲线 $y=f(x)$ 与 $y=L_{10}(x)$，可以看出在 $x=\pm5$ 附近，$L_{10}(x)$ 偏离 $f(x)$ 很远，例如 $L_{10}(4.8)\approx$

1.80，$f(4.8)\approx 0.04$. 这种现象称为"Runge 现象"。

2.8.1 图



由图知，Runge 考察的函数存在各阶导数. 这说明，即使对具有良好性质的函数，高次插值也未必有效. 另外，由于函数值的随机误差，或对不大光滑的函数，会使高阶插值公式产生大的误差. 因此，在实践中通常仅用分段低次插值.

2.8.2 分段线性插值

分段线性插值是用折线逼近函数，是最简单的分段低次插值.

用 f 表示区间$[a,b]$上的函数，引进记号

2.8.2 $C^k[a,b]=\{f \mid f^{(k)}$在$[a,b]$上连续$\}$

即 $C^k[a,b]$是$[a,b]$上具有连续 k 阶导数的函数全体所成之集合. 特别地，记

2.8.3 $C[a,b]=C^0[a,b]$

这是$[a,b]$上连续函数全体所成之集合. 以后可以用记号$f\in C^k[a,b]$表示 f 是$[a,b]$具有连续 k 阶导数的函数.

2.8.4 **定义** 设f是$[a,b]$上的函数，在节点 $a=x_0<x_1<\cdots<x_n=b$上的函数值为 $f_0,f_1,\cdots,f_n$. 记$h=\max\limits_{0\leqslant i\leqslant n-1} h_i$；$h_i=x_{i+1}-x_i$

$(i=0,1,\cdots,n-1)$. 如果函数I_h满足条件：

1° $I_h\in C[a,b]$；

2° $I_h(x_i)=f_i(i=0,1,\cdots,n)$；

3° 在每个子区间$[x_i,x_{i+1}]$ $(0\leqslant i\leqslant n-1)$上，$I_h$ 是次数不超过1 的多项式；

则称 I_h 是 f 的**分段线性插值函数**。

依据定义，I_h在子区间$[x_i,x_{i+1}]$上有

2.8.5 $$I_h(x)=f_i\frac{x-x_{i+1}}{x_i-x_{i+1}}+f_{i+1}\frac{x-x_i}{x_{i+1}-x_i}\quad(x_i\leqslant x\leqslant x_{i+1})$$

由此可得 I_h 在整个区间$[a,b]$上的表达式

2.8.6 $$I_h(x)=\sum_{i=0}^{n}f_il_i(x),$$

其中 l_i 是分段线性插值的基函数. l_i 的定义是

2.8.7 $$l_i(x)=\begin{cases}0, & x\notin[x_{i-1},x_{i+1}]\\ \dfrac{x-x_{i-1}}{x_i-x_{i-1}}, & x\in[x_{i-1},x_i]\ (i=0\text{时略去})\\ \dfrac{x-x_{i+1}}{x_i-x_{i+1}}, & x\in[x_i,x_{i+1}]\ (i=0\text{时略去})\end{cases}$$

下面的定理表明，分段线性插值能任意逼近连续函数.

2.8.8 定理 设 $f\in C[a,b]$，则当 $h\to0$ 时 I_h 一致收敛于f，即对任意一常数 $\varepsilon>0$，均存在相应的 $\delta>0$，只要 $h<\delta$，便有 $|f(x)-I_h(x)|<\varepsilon$，$\forall x\in[a,b]$

如果f 在$[a,b]$上存在二阶导数，利用线性插值余项的界 (2.2.15)，可得分段线性插值的余项 $R(x)=f(x)-I_h(x)$ 的一个界

2.8.9 $|R(x)| \leqslant \frac{Mh^2}{8}, \quad M = \max_{a \leqslant x \leqslant b} |f''(x)|$

这时定理（2.8.8）的结论是明显的。

2.8.3 分段三次 Hermite 插值

分段线性插值函数在节点处导数不存在。如果需要插值函数在节点处也可导，并且在节点处除了提供函数值外还提供了导数值，则可进行分段 Hermite 插值。

2.8.10 **定义** 设函数f在节点$a = x_0 < x_1 < \cdots < x_n = b$上的函数为$f_0, f_1, \cdots, f_n$，导数值为$f_0', f_1', \cdots, f_n'$。如果函数$I_h$满足条件：

1° $I_h \in C^1[a, b]$；

2° $I_h(x_i) = f_i$，$I_h'(x_i) = f_i' \quad (i = 0, 1, \cdots, n)$；

3° 在每个子区间$[x_i, x_{i+1}]$$(0 \leqslant i \leqslant n-1)$上，$I_h$是次数不超过 3 的多项式；

则称I_h是f的**分段Hermite插值函数**。

由两点三次插值公式（2.7.11），I_h在子区间$[x_i, x_{i+1}]$上有

2.8.11
$$I_h(x) = f_i \cdot \left(1 + 2\frac{x - x_i}{x_{i+1} - x_i}\right)\left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 + f_{i+1} \cdot \left(1 + 2\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)\left(\frac{x - x_i}{x_{i+1} - x_i}\right)^2 + f_i' \cdot (x - x_i)\left(\frac{x - x_{i+1}}{x_i - x_{i+1}}\right)^2 + f'_{i+1} \cdot (x - x_{i+1})\left(\frac{x - x_i}{x_{i+1} - x_i}\right)^2$$

由此可得I_h在整个区间$[a, b]$上的表达式

2.8.12
$$I_h(x) = \sum_{i=0}^{n}\left[f_i \alpha_i(x) + f_i' \beta_i(x)\right]$$

其中α_i，β_i是分段三次 Hermite 插值的基函数。α_i，β_i的定

义是

2.8.13 $$\alpha_i(x)=\begin{cases}0, & x\notin[x_{i-1},x_{i+1}]\\ \left(1+2\frac{x-x_i}{x_{i-1}-x_i}\right)\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2, & x\in[x_{i-1},x_i]\ (i=0\text{时略去})\\ \left(1+2\frac{x-x_i}{x_{i+1}-x_i}\right)\left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2, & x\in[x_i,x_{i+1}]\ (i=n\text{时略去})\end{cases}$$

2.8.14 $$\beta_i(x)=\begin{cases}0, & x\notin[x_{i-1},x_{i+1}]\\ (x-x_i)\left(\frac{x-x_{i-1}}{x_i-x_{i-1}}\right)^2, & x\in[x_{i-1},x_i]\ (i=0\text{时略去})\\ (x-x_i)\left(\frac{x-x_{i+1}}{x_i-x_{i+1}}\right)^2, & x\in[x_i,x_{i+1}]\ (i=n\text{时略去})\end{cases}$$

可以证明，若$f\in C[a,b]$，$h=\max\limits_{0\leqslant i\leqslant n-1}(x_{i+1}-x_i)$，则当$h\to 0$时，分段三次 Hermite 插值函数 I_h一致收敛于f.

2.9

2.9 样条插值

2.9.1 样条函数

样条插值也是用分段低次多项式去逼近函数. 在分段低次插值中，分段线性插值函数在节点处不可导，分段三次 Hermite 插值函数有连续一阶导数，光滑性也较差，而且需要提供各个节点处的导数值. 样条插值则能较好地适应对光滑性的不同需求，并且只需在插值区间端点提供某些导数信息.

样条插值是用样条函数去逼近函数.

2.9.1 **定义** 设在区间$[a,b]$上给定一个分划

$a=x_0<x_1<\cdots<x_n=b$

如果函数 S 满足条件：

1° $S\in C^{k-1}[a,b]$；

2° 在每个子区间$[x_i,x_{i+1}]$$(0\leqslant i\leqslant n-1)$上，

S 是次数不超过 k 的多项式；

则称 S 是以 x_0，x_1，…，x_n 为节点的k次样条函数（Spline of Degree k）。

近几十年来，函数逼近在理论研究和实际应用中均获得重大的进展，其中非常流行和十分活跃的分支是样条函数。在飞机、船舶、汽车等外形设计的工程问题中，在数值微分、数值积分、微分方程和积分方程的数值解法，以及观测和实验数据的处理等方面，样条函数都有着重要的应用。

2.9.2 B 样条

B 样条是一类基本的样条函数。

2.9.2 定义 记

$$u_+^m=\begin{cases} u^m, & 当u\geqslant 0 \\ 0, & 当u<0 \end{cases} \quad (m=1,2,\cdots)$$

u_+^m 称为m次半截单项式.并规定

$$u_+^0=\begin{cases} 1, & 当u>0 \\ \dfrac{1}{2}, & 当u=0 \\ 0, & 当u<0 \end{cases}$$

利用定义（2.9.2）可以直接给出B 样条的定义。

2.9.3 定义 设 Ω_k 是$(-\infty,\ \infty)$上的函数，

$$\Omega_k(x)=\frac{1}{k!}\sum_{j=0}^{k+1}(-1)^j\binom{k+1}{j}\left(x+\frac{k+1}{2}-j\right)_+^k$$

Ω_k 称为 k 次 B 样条或基本样条函数。

可以看出，Ω_k 是一个分段 k 次多项式，k 阶导数间断点为

$$x_j=j-\frac{k+1}{2}\quad (j=0,1,\cdots,k+1)$$

因此，Ω_k 是以 x_0，x_1，…，x_{k+1} 为节点的 k 次样条函数。Ω_k 在整个实数轴 $(-\infty, \infty)$ 上有定义，但是容易证明，它在区间 $[x_0, x_{k+1}]$ 即 $\left[-\frac{k+1}{2}, \frac{k+1}{2}\right]$ 之外恒为零。下面是几个低次B 样条的表达式及相应的图形：

2.9.4 0 至 3 次B 样条表达式

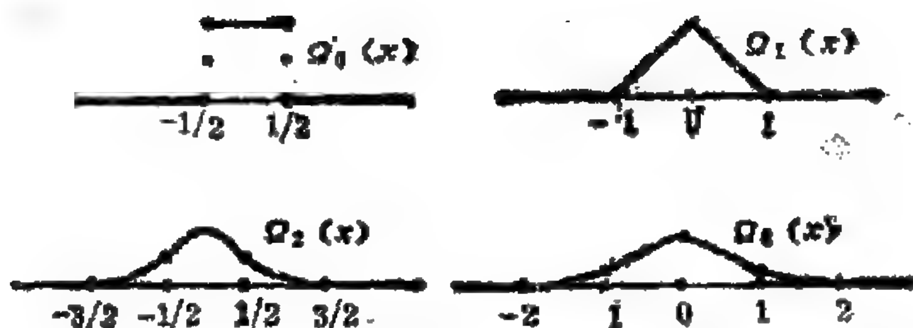
$$\Omega_0(x)=\begin{cases}0 & \left(|x|>\frac{1}{2}\right)\\ 1 & \left(|x|<\frac{1}{2}\right)\\ \frac{1}{2} & \left(|x|=\frac{1}{2}\right)\end{cases}$$

$$\Omega_1(x)=\begin{cases}0 & (|x|\geqslant 1)\\ 1-|x| & (|x|<1)\end{cases}$$

$$\Omega_2(x)=\begin{cases}0 & \left(|x|>\frac{3}{2}\right)\\ -x^2+\frac{3}{4} & \left(|x|<\frac{1}{2}\right)\\ \frac{1}{2}x^2-\frac{3}{2}|x|+\frac{9}{8} & \left(\frac{1}{2}\leqslant|x|\leqslant\frac{3}{2}\right)\end{cases}$$

$$\Omega_3(x)=\begin{cases}0 & (|x|\geqslant 2)\\ \frac{1}{2}|x|^3-x^2+\frac{2}{3} & (|x|\leqslant 1)\\ -\frac{1}{6}|x|^3+x^2-2|x|+\frac{4}{3} & (1<|x|<2)\end{cases}$$

2.9.5 图



从图（2.9.5）可以看出，随次数的增加，B 样条的曲线越来越光滑，节点（图中的圆点）越来越多。实际上，定义（2.9.3）中 Ω_k 的表达式是由下述递推公式得出的：

2.9.6 $$\Omega_i(x)=\int_{x-\frac{1}{2}}^{x+\frac{1}{2}}\Omega_{i-1}(x)dx \quad (i=1,2,\cdots)$$

其中 Ω_0 已在（2.9.4）中给出。由于积分运算可以使间断函数变得连续，使粗糙的函数曲线变得光滑，因此，次数越高的B 样条其曲线越光滑。

在样条函数的研究中，无论是理论分析还是实际数值计算，B 样条 Ω_k 都有它的独特作用。

2.9.3 三次样条插值问题的提法

三次样条是应用最广泛的样条，它有明确的力学意义。早期绘图员在制图时，用一种富有弹性的细长木条，称为样条，把它用压铁固定在若干样点上后画出的曲线称为样条曲线。这种样条曲线在数学上表现为符合定义（2.9.1）的三次样条函数。

2.9.7 **定义** 设在区间$[a, b]$上给定一个分划

$a=x_0<x_1<\cdots<x_n=b$

S 是以 x_0，x_1，$\cdots$，x_n 为节点的三次样条函数。如果 S 满足插值条件

2.9.8 $S(x_i)=y_i \quad (i=0,\ 1,\ \cdots,\ n)$

则称 S 为**三次样条插值函数**.

由定义，关于 S 的可供利用的条件有 $4n-2$ 个，其中插值条件包含了 $n+1$ 个，另由 $S\in C^2[a,\ b]$，节点处的连续性条件有 $3n-3$ 个：

2.9.9
$$\begin{cases} S(x_i-0)=S(x_i+0) \\ S'(x_i-0)=S'(x_i+0) \quad (i=1,2,\cdots,n-1) \\ S''(x_i-0)=S''(x_i+0) \end{cases}$$

但是，S 由 n 段次数不超过 3 的多项式组成，共有 $4n$ 个待定参数。因此，为了确定 S，还缺少 2 个条件。通常是在区间$[a,\ b]$的端点 $a=x_0$，$b=x_n$ 上各附加一个条件。在端点上的条件称为**边界条件**。插值条件(2.9.8)中已包含 2 个边界条件。常见的附加边界条件有三种，并由此提出以下三个类型的插值问题。

问题 I　求三次样条插值函数 S，满足附加边界条件

2.9.10 $S'(x_0)=y_0',\ S'(x_n)=y_n'$

问题 II　求三次样条插值函数 S，满足附加边界条件

2.9.11 $S''(x_0)=y_0'',\ S''(x_n)=y_n''$

其特殊情况

2.9.12 $S''(x_0)=S''(x_n)=0$

称为**自然边界条件**。

问题 III　当被插函数是以x_n-x_0为周期的周期函数时，要寻求三次样条 S，满足内部节点处的条件

$S(x_i)=y_i \quad (i=1,\ 2,\ \cdots,\ n-1)$

及边界条件

2.9.13 $S(x_0)=S(x_n)=y_0,\ S^{(k)}(x_0+0)=S^{(k)}(x_n-0)\ (k=1,2)$

这是周期型问题，所寻求的样条函数 S 称为**周期样条函数**(Periodic Spline Function)。

2.9.14 定理 插值问题Ⅰ、Ⅱ、Ⅲ的解均存在唯一。

因此，问题是如何具体去构造三次样条插值函数S。

2.9.4 均匀分划的三次样条插值函数

设分划是均匀的，即

2.9.15 $x_i=a+ih \quad (i=0,1,\cdots,n), \quad h=\dfrac{b-a}{n}$

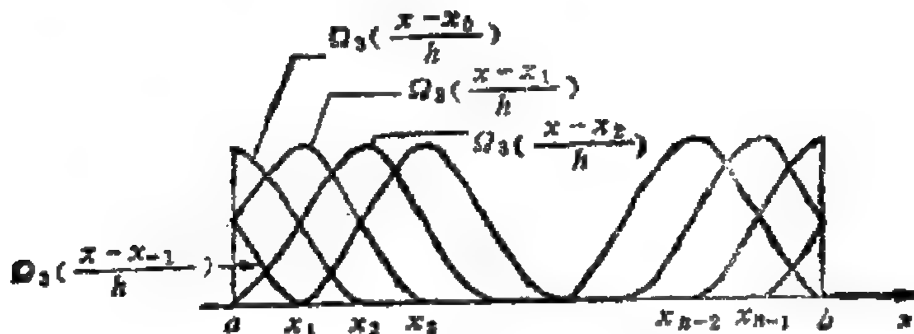
这时，有一组三次B样条

2.9.16 $$\Omega_3\left(\frac{x-x_i}{h}\right)=\Omega_3\left(\frac{x-x_0}{h}-i\right)$$

$$(i=-1,0,1,\cdots,n+1)$$

它们中的每一个在$[a, b]$上都不恒等于零，其它i所对应的B样条在$[a, b]$则恒为零（见图2.9.17）。

2.9.17 图



（2.9.16）中共有$n+3$个B样条，考虑以它们的线性组合作为插值函数S，

2.9.18 $$S(x)=\sum_{j=-1}^{n+1} c_j\Omega_3\left(\frac{x-x_0}{h}-j\right)$$

其中$c_j(-1\leqslant j\leqslant n+1)$是待定常数。这样定义的$S$明显是以$x_0, x_1, \cdots, x_n$为节点的3次样条函数，（2.9.9）中的

$3n-3$ 个条件能自动地满足。因此剩下的插值条件(2.9.8)及附加边界条件共包含 $n+3$ 个条件恰好可确定 c_{-1}, c_0,…, c_{n+1}。

对问题Ⅰ 根据条件 (2.9.8) 及 (2.9.10)，有

2.9.19
$$\begin{cases} S'(x_0)=\dfrac{1}{h}\sum\limits_{j=-1}^{n+1} c_j\Omega_3'(-j)=y_0' \\ S(x_i)=\sum\limits_{j=-1}^{n+1} c_j\Omega_3(i-j)=y_i \qquad (i=0,1,\cdots,n) \\ S'(x_n)=\dfrac{1}{h}\sum\limits_{j=-1}^{n+1} c_j\Omega_3'(n-j)=y_n' \end{cases}$$

利用 (2.9.4) 中 Ω_3 的表达式，可得

$$\Omega_3(0)=\frac{2}{3},\quad \Omega_3(\pm 1)=\frac{1}{6},\quad \Omega_3(x)=0 \qquad (|x|\geqslant 2)$$

$$\Omega'_3(0)=0,\quad \Omega_3'(\pm 1)=\mp\frac{1}{2},\quad \Omega'_3(x)=0 \qquad (|x|\geqslant 2)$$

于是 (2.9.19) 的具体形式是

2.9.20
$$\begin{cases} \dfrac{-c_{-1}+c_1}{2h}=y_0' \\ \dfrac{1}{6}c_{i-1}+\dfrac{2}{3}c_i+\dfrac{1}{6}c_{i+1}=y_i \qquad (i=0,1,\cdots,n) \\ \dfrac{-c_{n-1}+c_{n+1}}{2h}=y_n' \end{cases}$$

最后归结为解线性代数方程组：

2.9.21 $AC=F$

其中

$$A=\begin{pmatrix} 4 & 2 & & & & \\ 1 & 4 & 1 & & & \\ & 1 & 4 & 1 & & \\ & & & \cdots & & \\ & & & 1 & 4 & 1 \\ & & & & 2 & 4 \end{pmatrix}$$

是 $n+1$ 阶具有严格对角优势的三对角矩阵，而

$$F=\begin{pmatrix} 6y_0+2hy_0' \\ 6y_1 \\ 6y_2 \\ \vdots \\ 6y_{n-1} \\ 6y_n-2hy_n' \end{pmatrix}, \quad C=\begin{pmatrix} c_0 \\ c_1 \\ c_2 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix}$$

还有关系式

2.9.22 $$\begin{cases} c_{-1}=c_1-2hy_0' \\ c_{n+1}=c_{n-1}+2hy_n' \end{cases}$$

对以三对角矩阵为系数矩阵的线性代数方程组有方便的解法(见第六章).从(2.9.21)解出 c_0，c_1，…，c_n，再从(2.9.22)算出 c_{-1}，c_{n+1}，然后代入（2.9.18)，便得所求样条插值函数 S.

对问题Ⅱ 由条件（2.9.8）及（2.9.11)，可归结出如下代数方程组：

2.9.23 $AC=F$

其中

$$A=\begin{bmatrix} 4 & 1 & & & & & \\ 1 & 4 & 1 & & & & \\ & 1 & 4 & 1 & & & \\ & & & \cdots & & & \\ & & & & 1 & 4 & 1 \\ & & & & & 1 & 4 \end{bmatrix}$$

是 $n-1$ 阶严格对角占优的对称三对角矩阵，而

$$F=\begin{bmatrix} 6y_1-y_0+\frac{h^2}{6}y_0'' \\ 6y_2 \\ 6y_3 \\ \vdots \\ 6y_{n-2} \\ 6y_{n-1}-y_n+\frac{h^2}{6}y_n'' \end{bmatrix}, C=\begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-2} \\ c_{n-1} \end{bmatrix}$$

还有关系式

$$2.9.24 \quad \begin{cases} c_{-1}=2c_0-c_1+h^2y_0'' \\ c_0=y_0-\frac{h^2}{6}y_0'' \\ c_n=y_n-\frac{h^2}{6}y_n'' \\ c_{n+1}=2c_n-c_{n-1}+h^2y_n'' \end{cases}$$

从（2.9.23）解出 c_1，c_2，…，c_{n-1}，再从（2.9.24）算出 c_{-1}，c_0，c_n，c_{n+1}。

对问题Ⅱ　由条件（2.9.8）及（2.9.13）可以推出

2.9.25　$c_{n+1}=c_1$，$c_0=c_n$，$c_{-1}=c_{n-1}$

及线性代数方程组

2.9.26　$AC=F$

其中

$$
A=\begin{pmatrix}
4 & 1 & & & & & \\
1 & 4 & 1 & & & & \\
 & 1 & 4 & 1 & & & \\
 & & & \cdots & & & \\
 & & & & 1 & 4 & 1 \\
 & & & & & 1 & 4
\end{pmatrix}
$$

是 n 阶矩阵，而

$$
F=\begin{pmatrix} 6y_1 \\ 6y_2 \\ 6y_3 \\ \vdots \\ 6y_{n-1} \\ 6y_0 \end{pmatrix}, \quad C=\begin{pmatrix} c_1 \\ c_2 \\ c_3 \\ \vdots \\ c_{n-1} \\ c_n \end{pmatrix}
$$

从（2.9.26）解出 c_1，c_2，…，c_n，由（2.9.25）可直接确定 c_{n+1}，c_0，c_{-1}。

2.9.5 任意分划的三次样条插值函数

假设分划是任意的，构造三次样条插值函数 S 的一种方法如下，令

2.9.27 $S''(x_i)=M_i \quad (i=0, 1, \cdots, n)$

由于 S 是分段次数不超过 3 的多项式，而且二阶导数 S'' 连续，可知 S'' 是折线函数，因此在任意取定的子区间 $[x_i, x_{i+1}]$ 上有

2.9.28 $S''(x)=M_i\dfrac{x_{i+1}-x}{h_i}+M_{i+1}\dfrac{x-x_i}{h_i}$

其中 $h_i=x_{i+1}-x_i$。将（2.9.28）积分两次，并利用插值条件 $S(x_i)=y_i$，$S(x_{i+1})=y_{i+1}$

确定两个积分常数，得到

2.9.29 $$S(x)=M_i\frac{(x_{i+1}-x)^3}{6h_i}+M_{i+1}\frac{(x-x_i)^3}{6h_i}+\left(\frac{y_i}{h_i}-\frac{M_ih_i}{6}\right)(x_{i+1}-x)+\left(\frac{y_{i+1}}{h_i}-\frac{M_{i+1}h_i}{6}\right)(x-x_i)\qquad (x\in[x_i,x_{i+1}])$$

对 S 求导，又得

2.9.30 $$S'(x)=-M_i\frac{(x_{i+1}-x)^2}{2h_i}+M_{i+1}\frac{(x-x_i)^2}{2h_i}+\frac{y_{i+1}-y_i}{h_i}-\frac{M_{i+1}-M_i}{6}h_i\qquad (x\in[x_i,x_{i+1}])$$

利用（2.9.9）中的条件

$S'(x_i-0)=S'(x_i+0)\qquad (i=1,\ 2,\ \cdots,\ n-1)$

并根据在子区间$[x_{i-1},\ x_i]$及$[x_i,\ x_{i+1}]$上 S' 的两种表达式，可得方程

2.9.31 $$\mu_iM_{i-1}+2M_i+\lambda_iM_{i+1}=d_i\qquad (i=1,\ 2,\ \cdots,\ n-1)$$

其中

2.9.32 $$\begin{cases}\mu_i=\dfrac{h_{i-1}}{h_{i-1}-h_i}\\ \lambda_i=1-\mu_i\\ d_i=\dfrac{6}{h_{i-1}+h_i}\left(\dfrac{y_{i+1}-y_i}{h_i}-\dfrac{y_i-y_{i-1}}{h_{i-1}}\right)\end{cases}$$

（2.9.31）是关于 $n+1$ 个未知数 M_0，M_1，…，M_n的 $n-1$ 个方程，尚须附加边界条件，补充两个方程。

对问题Ⅰ 由（2.9.10）并利用（2.9.30）可导出如下两个方程：

$$
2.9.33 \quad \begin{cases} 2M_0 + M_1 = \dfrac{6}{h_0}\left(\dfrac{y_1 - y_0}{h_0} - y_0'\right) \\ M_{n-1} + 2M_n = \dfrac{6}{h_{n-1}}\left(y_n' - \dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{cases}
$$

对问题Ⅱ　由条件（2.9.11）直接有

$$
2.9.34 \quad \begin{cases} M_0 = y_0'' \\ M_n = y_n'' \end{cases}
$$

对问题Ⅲ　由条件（2.9.13）有

$y_n = y_0$，

而且可导出两个方程：

$$
2.9.35 \quad \begin{cases} M_n = M_0 \\ \lambda_n M_1 + \mu_n M_{n-1} + 2M_n = d_n \end{cases}
$$

其中

$$
2.9.36 \quad \begin{cases} \mu_n = \dfrac{h_{n-1}}{h_0 + h_{n-1}} \\ \lambda_n = 1 - \mu_n \\ d_n = \dfrac{6}{h_0 + h_{n-1}}\left(\dfrac{y_1 - y_0}{h_0} - \dfrac{y_n - y_{n-1}}{h_{n-1}}\right) \end{cases}
$$

由（2.9.31）与（2.9.33），或（2.9.31）与（2.9.34），或（2.9.31）与（2.9.35），分别构成具有三对角非奇异系数矩阵的方程组，可以解出 M_0，M_1，…，M_n，并由（2.9.29）与（2.9.30）确定样条插值函数及其导函数。这样，插值问题Ⅰ、Ⅱ、Ⅲ均得到解决。

构造任意分划的三次样条插值函数存在不同的方法。上述方法，M_i相应于力学中细梁在 x_i 处截面的弯矩，每一个方程中又至多出现相邻的三个 M_i，通常称为**三弯矩法**。

2.9.6 三次样条插值的收敛性

利用三次样条插值，不会出现 Lagrange 插值时的那种“Runge 现象”。为了提高插值的精确度，样条插值可用增加节点的办法来实现，而 Lagrange 插值增加节点会导致多项式次数的增高，常引起计算的不稳定。下面仍取第 8 节中 Runge 提供的函数为数值计算的实例。

2.9.37 例 设 $f(x)=\dfrac{1}{1+x^2}$，$x\in[-5,5]$，节点为

$x_i=-5+i \quad (i=0,\ 1,\ \cdots,\ 10)$

求三次样条插值函数S，满足插值条件

$S(x_i)=f(x_i) \quad (i=0,\ 1,\ \cdots,\ 10)$

及附加边界条件

$S'(-5)=f'(-5)$，$S'(5)=f'(5)$

即对函数 f 求样条插值问题 I 的解。

解 表(2.9.38)给出了 S 的数值结果，为了比较，表中还列出了 f 及 Lagrange 插值多项式 L_{10} 的相应结果。L_{10} 的几何曲线已在图 (2.8.4) 中给出。可以看出，S 能很好近似 f。

2.9.38 表

x	$\dfrac{1}{1+x^2}$	$S(x)$	$L_{10}(x)$
−5.0	0.03846	0.03846	0.03846
−4.8	0.04160	0.03758	1.80438
−4.3	0.05131	0.04842	0.88808
−4.0	0.05882	0.05882	0.05882
−3.8	0.06477	0.06556	−0.20130
−3.3	0.08410	0.08426	−0.10832
−3.0	0.10000	0.10000	0.10000
−2.8	0.11312	0.11366	0.19837

续表

x	$\frac{1}{1+x^2}$	$S(x)$	$L_{10}(x)$
−2.3	0.15898	0.16115	0.24145
−2.0	0.20000	0.20000	0.20000
−1.8	0.23585	0.23154	0.18878
−1.3	0.37175	0.36133	0.31650
−1.0	0.50000	0.50000	0.50000
−0.8	0.60975	0.62420	0.64316
−0.3	0.91743	0.92754	0.94090
0.0	1.00000	1.00000	1.00000

对于 $f\in C[a,b]$，记

2.9.39　$\|f\|_\infty=\max\limits_{a\leqslant x\leqslant b}|f(x)|$

称为函数 f 的∞－**范数**。

(2.9.40) 给出了关于三次样条插值收敛性的一个结论。

2.9.40　**定理**　设 $f\in C^4[a,b]$，S 为问题Ⅰ或问题Ⅱ的插值函数，令

$$\beta=\frac{h}{\min\limits_{0\leqslant i\leqslant n-1} h_i},\quad h=\max\limits_{0\leqslant i\leqslant n-1} h_i,\quad h_i=x_{i+1}-x_i$$

$$(i=0,1,\cdots,n-1)$$

β 称为分划比。则有估计式

2.9.41　$\|f^{(k)}-S^{(k)}\|_\infty\leqslant c_k\|f^{(4)}\|_\infty h^{4-k}\quad(k=0,1,2,3)$

其中

$$c_0=\frac{5}{384},\ c_1=\frac{1}{24},\ c_2=\frac{3}{8},\ c_3=\frac{\beta+\beta^{-1}}{2}$$

这个定理中系数 c_0，c_1，c_2 均与分划比无关，仅 c_3 与分划比 β 有关。因此，当 $h\to0$ 时，(2.9.41) 表明，样条插值函数 S 及其一阶导数 S'，二阶导数 S''，分别一致收敛于 f，

f'，f''。若分划比在加密过程中能保证

$$0<m\leqslant\beta\leqslant M$$

这里 m 与 M 是常数，则还有 S''' 一致收敛于 f'''。

2.10

2.10 反插值

2.10.1 插值与反插值

设函数 f 的离散数据为

2.10.1 (x_i, y_i)，$y_i=f(x_i)$ $(i=0, 1, \cdots, n)$

插值的目的是在 x_0，x_1，…，x_n 之间给定了自变量 x 的值后，要去求函数 f 的近似值，其途径是构造插值多项式。不同的构造方法，就是不同的插值法。与此相反，反插值的目的是在 y_0，y_1，…，y_n之间给定了函数 f 的值后，要去求自变量 x 的近似值，其途径仍是利用插值法。

反插值有两种处理方式：一种是直接利用函数 f 的插值多项式；另一种是在假设 f^{-1} 是存在的前提下，构造f 的反函数 f^{-1} 的插值多项式。

因为反插值归结为求满足

2.10.2 $f(x)=c$

的 x 的近似值，这里 c 是在 y_0，y_1，…，y_n 之间的某个值。如果 f 的反函数 f^{-1} 存在，则

$$x=f^{-1}(c)$$

反插值就是求f^{-1} 在 c 上的近似值。当 $c=0$ 时，反插值就是求函数 f 的近似零点，或者说是求方程 $f(x)=0$ 的近似根。所以反插值有明显的意义。

2.10.2 利用函数的插值多项式反插

考虑在 x_0 与 x_1 之间进行反插值。假设 c 是$y_0=f(x_0)$与 $y_1=f(x_1)$之间的一个值，则当 f 连续时，在x_0 与 x_1 之间必存在 x，使得 $f(x)=c$，寻求 x 的近似值，一般只利用 f 的

低次插值多项式。

从离散数据算出 f 的均差表，如果均差表中的二阶均差实际上可看为零，则 f 在 x_0 与 x_1 之间可用一次插值多项式 p_1近似代替，

2.10.3 $$p_1(x)=f(x_0)+f[x_0, x_1](x-x_0)$$

令 $p_1(x)=c$，从（2.10.3）解出 x，即得所要求的近似值。

如果均差表中到三阶均差才实际上可看为零，则 f 在x_0 与 x_1 之间可用二次插值多项式 p_2 近似代替，

2.10.4 $$p_2(x)=f(x_0)+f[x_0, x_1](x-x_0)+f[x_0, x_1, x_2](x-x_0)(x-x_1)$$

这时，令 $p_2(x)=c$，要从（2.10.4）去解出 x，一般用逐次逼近法：先取 $x^{(0)}$，使得满足

$$f(x_0)+f[x_0, x_1](x^{(0)}-x_0)=c$$

即

$$x^{(0)}=x_0+\frac{c-f(x_0)}{f[x_0, x_1]}$$

再求 $x^{(1)}$，使得满足

$$f(x_0)+f[x_0, x_1](x^{(1)}-x_0)+f[x_0, x_1, x_2](x^{(0)}-x_0)(x^{(0)}-x_1)=c$$

即

$$x^{(1)}=x_0+\frac{c-f(x_0)}{f[x_0, x_1]}-\frac{f[x_0, x_1, x_2]}{f[x_0, x_1]}(x^{(0)}-x_0)(x^{(0)}-x_1)$$

然后可利用迭代公式

2.10.5 $$x^{(k)}=x^{(0)}-\frac{f[x_0, x_1, x_2]}{f[x_0, x_1]}(x^{(k-1)}-x_0)(x^{(k-1)}-x_1)$$

反复进行迭代，直至 $x^{(k)}$与$x^{(k-1)}$在所要求的精确度下相等为止。

如果在均差表中三阶均差实际上还不是零，则可用更高

次的插值多项式.

对于等距节点情形，可改用差分形式的插值公式. 设

$x_i = x_0 + ih \quad (i=0, 1, \cdots, n)$

且

$x = x_0 + th \quad (0<t<1)$

则迭代公式（2.10.5）可改写成

2.10.6 $t^{(k)} = t^{(0)} - \dfrac{\Delta^2 y_0}{2\Delta y_0} t^{(k-1)}(t^{(k-1)} - 1) \quad (k=1,2,\cdots)$

其中

$t^{(0)} = \dfrac{c - y_0}{\Delta y_0}$

2.10.7 例　已知 $f(x)=\sin x$ 的函数表如下：

x_i	$y_i = \sin x_i$	x_i	$y_i = \sin x_i$
1.72	0.9888898	1.78	0.9781966
1.74	0.9857192	1.80	0.9738476
1.76	0.9821543	1.82	0.9691091

求 $\sin^{-1}0.9800000$ 之值.

解　取 $x_0=1.76$，$y_0=0.9821543$. 经计算得

$\Delta y_0 = -0.0039577$，$\Delta^2 y_0 = -0.0003913$

$t^{(0)} = \dfrac{0.9800000 - 0.9821543}{-0.0039577} = 0.5443313$

采用二次插值多项式，将 $t^{(0)}, \Delta y_0, \Delta^2 y_0$ 的值代入（2.10.6），得到迭代公式

$t^{(k)} = 0.5443313 - 0.04943527 t^{(k-1)}(t^{(k-1)} - 1)$

$(k=1,2,\cdots)$

由此公式算出

$t^{(1)} = 0.5565926$，$t^{(2)} = 0.5565318$，$t^{(3)} = 0.5565321$

从$t^{(3)}$得到

$$\sin^{-1}0.9800000 \approx x^{(3)} = x_0 + t^{(3)}h$$
$$= 1.76 + 0.5565321 \times 0.02 = 1.771131$$

这个值准确到第六位小数。

2.10.3 构造反函数的插值多项式

设函数 f 在点 x_0，x_1，…，x_n 上的值为

y_0，y_1，…，y_n。

若 f 的反函数 f^{-1} 存在，则 f^{-1} 在点y_0，y_1，…，y_n 上的值为

x_0，x_1，…，x_n

这样，反函数 f^{-1} 以y_0，y_1，…，y_n 为节点的插值多项式是

2.10.8 $$p(y) = f^{-1}(y_0) + f^{-1}[y_0, y_1](y-y_0) + f^{-1}[y_0, y_1, y_2](y-y_0)(y-y_1) + \cdots + f^{-1}[y_0, y_1, \cdots, y_n](y-y_0)(y-y_1)\cdots(y-y_{n-1})$$

由于这是利用f的离散数据来建立f^{-1}的插值公式，y_0，y_1，…，y_n一般不会是等距分布，因此上面写出的是均差形式的公式。

对 y_0，y_1，…，y_n 确定的区间内的任何值c，可由(2.10.8)算出$f^{-1}(c)$的近似值，或者算出方程$f(x)=c$的近似解。

2.10.9 **例** 设 $f(x)=x^3-3x^2-x+9$，已知 f 的下列函数值：

x	-1.3	-1.4	-1.5	-1.6	-1.7
$f(x)$	3.033	1.776	0.375	-1.176	-2.883

求方程$f(x)=0$ 在区间$[-1.7, -1.3]$上的根的近似值。

解 f的反函数 f^{-1} 的均差如下：

y_i	$f^{-1}(y_i)$	$f[x_0,x_i]$	$f[x_0,x_1,x_i]$	$f[x_0,x_1,x_2,x_i]$	$f[x_0,\cdots,x_4]$
3.033	−1.3				
1.776	−1.4	0.0795545			
0.375	−1.5	0.0752445	0.0030763		
−1.176	−1.6	0.0712758	0.0028044	0.0001753	
−2.883	−1.7	0.0676133	0.0025630	0.0001575	0.0000104

由均差表得出f^{-1}的插值多项式是

$$
\begin{aligned}
p(y) = & -1.3+0.0795545(y-3.033) \\
& +0.0030763(y-3.033)(y-1.776) \\
& +0.0001753(y-3.033)(y-1.776)(y-0.375) \\
& +0.0000104(y-3.033)(y-1.776)(y-0.375) \\
& (y+1.176)
\end{aligned}
$$

经计算可得方程$f(x)=0$在$[-1.7,\ -1.3]$上的根x的近似值

$$x=f^{-1}(0)\approx p(0)=-1.525097$$

根x的实际七位准确值是-1.525102。

3

第3章 函数逼近与曲线拟合

3.1

3.1 一致逼近

在区间$[a, b]$上给定一个连续函数$f(\cdot)$，记作$f\in C[a, b]$，用简单函数 $p(\cdot)$去近似$f(\cdot)$，就是函数逼近要研究的问题．通常，$p(\cdot)$可取为多项式、有理分式或三角多项式，但由于计算机上只能做四则运算，所以，如果要利用逼近函数计算$f(x)$的值，则取多项式与有理分式逼近，其实际意义更大．度量逼近误差的标准可以各不相同，但比较重要的只有两种，其对应的函数逼近称为一致逼近与平方逼近．

设$f\in C[a, b]$，用 H_n 记所有次数$\leqslant n$ 的多项式集合，在区间$[a, b]$上用 $p_n\in H_n$逼近 $f(\cdot)$．其误差记作

3.1.1 $$\|f-p_n\|_\infty=\max_{a\leqslant x\leqslant b}|f(x)-p_n(x)|$$

如果$\lim\limits_{n\to\infty}\|f-p_n\|_\infty=0$，则称$p_n(\cdot)$在$[a, b]$上一致收敛到$f(\cdot)$．$p_n(\cdot)$称为$f(\cdot)$在$[a, b]$上的**一致逼近** (Uniform Approximation)多项式．对函数$f\in C[a, b]$，记

$$\|f\|_\infty=\max_{a\leqslant x\leqslant b}|f(x)|$$

称为f 的**最大范数** (Maximun Norm)或称为∞－范数。它满足范数的三条性质：

3.1.2 $$\begin{cases} 1^\circ \quad \|f\|\geqslant 0, \quad \|f\|=0 \text{ 当且仅当} f\equiv 0; \\ 2^\circ \quad \|af\|=|a|\,\|f\|, \quad \forall f\in C[a, b],\ a\in R^1; \\ 3^\circ \quad \|f+g\|\leqslant\|f\|+\|g\|, \quad \forall f,g\in C[a,b]. \end{cases}$$

按照这种度量标准，对任何$f\in C[a, b]$，总可找到一个多项式$p_n(\cdot)$，使 $\|f-p_n\|_\infty\leqslant\varepsilon$ (任意给定正数)，从而有下

面的重要定理。

3.1.3 定理 设$f\in C[a,b]$，对给定的$\varepsilon>0$，总存在一个多项式$p_n(x)$，使对$\forall x\in[a,b]$，有$|f(x)-p_n(x)|\leqslant\varepsilon$，这就是维尔斯特拉斯（Weierstrass)定理。

实际上，只要构造 Бернштейн（伯恩斯坦）多项式

3.1.4 $$B_n(f,x)=\sum_{k=0}^{n}\binom{n}{k}f\left(\frac{k}{n}\right)x^k(1-x)^{n-k}\quad(0\leqslant x\leqslant 1)$$

其中$\binom{n}{k}\ \frac{n(n-1)\cdots(n-k+1)}{k!}$取$[a,b]=[0,1]$，$p_n(x)=B_n(f,x)$，即可证明$\lim\limits_{n\to\infty}p_n(x)=f(x)$对$\forall x\in[0,1]$一致成立。这就表明任何连续函数总存在一致逼近多项式，但用$f(x)$的 Бернштейн 多项式逼近$f(x)$，收敛很慢。如果$f(x)$在$[a,b]$上的$n+1$阶导数连续，取$x_0\in(a,b)$则由 Taylor（泰勒）展开可知

$$f(x)=f(x_0)+f'(x_0)(x-x_0)+\cdots+\frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n+R_{n+1}(x)$$

记它的n部份和为$p_n(x)$，则有

3.1.5 $$p_n(x)=f(x_0)+f'(x_0)(x-x_0)+\cdots+\frac{1}{n!}f^{(n)}(x_0)(x-x_0)^n$$

3.1.6 $$R_{n+1}(x)=\frac{1}{(n+1)!}f^{(n+1)}(\xi)(x-x_0)^{n+1}$$

（ξ在x与x_0之间）

由于对$\forall x\in[a,b]$，有

$$\lim_{n\to\infty}|R_{n+1}(x)|=0$$

即$\lim\limits_{n\to\infty}\|f-p_n\|_\infty=0$，故$p_n(x)$是$f(x)$的一致逼近多项式。

3.1.7 例 对$f(x)=e^x$，用 Taylor 展开求$[-1, 1]$上的四阶一致逼近多项式。

解 取$x_0=0$，得

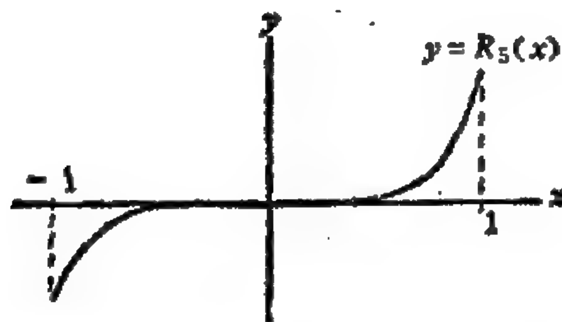
$$p_4(x)=1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4$$

$$R_5(x)=e^x-p_4(x)=\frac{e^{\xi}}{120}x^5 \quad (\xi\text{在}x\text{与}0\text{之间})$$

$$\| e^x-p_4(x) \|_\infty \leqslant \frac{e}{120} \doteq 0.0226$$

在区间$[-1, 1]$上用$p_4(x)$逼近e^x的误差分布见图（3.1.8），它在$x_0=0$附近误差较小，但在区间两端误差很大，表明误差分布极不均匀。

3.1.8 图 $p_4(x)\approx e^x$的误差曲线



一致逼近多项式并非唯一的，而且，当精度要求较高时，多项式次数n可能很大，因此为求得使误差(3.1.1)取得最小的逼近多项式，就要研究最佳一致逼近问题。

3.2

3.2 最佳一致逼近

在次数不超过n的多项式集合H_n中求$p_n(\cdot)$，使它与$f\in C[a, b]$的误差

$$\|f-p_n\|_\infty=\max_{a\leqslant x\leqslant b}|f(x)-p_n(x)|=\min$$

这就是最佳一致逼近问题。

3.2.1 **定义** 给定$f\in C[a,b]$，若存在$p_n^*\in H_n$，使

3.2.2 $\Delta(f,p_n^*)=\|f-p_n^*\|_\infty=\inf\limits_{p_n\in H_n}\|f-p_n\|_\infty=E_n$

则称p_n^*是f在$[a,b]$上的**最佳一致逼近**（Minimax Approximation）**多项式**，$\Delta(f,p_n^*)$称为**最佳偏差**（Minimax Error）。它等于最小偏差值E_n。

理论上已证明，对任何$f\in C[a,b]$，都存在唯一的$p_n^*\in H_n$，使（3.2.2）成立。实际上在集合H_n中每一元素$p_n\in H_n$都对应一个偏差$\Delta(f,p_n)$，由于$\Delta(f,p_n)=\|f-p_n\|_\infty\geqslant 0$，故集合$\{\Delta(f,p_n)\}$有下界，从而有下确界$E_n=\inf\limits_{p_n\in H_n}\Delta(f,p_n)$。如果存在$p_n^*\in H_n$使$\|f-p_n^*\|_\infty=E_n$，$p_n^*$就是所要求的最佳一致逼近多项式。Чебышев（切比雪夫）对最佳一致逼近多项式的特性，给出了下面的重要定理。

3.2.3 **定理**（Чебышев） 设$f\in C[a,b]$，$n\geqslant 0$。$p_n^*(\cdot)\in H_n$是f在$[a,b]$上的最佳一致逼近多项式的充分必要条件是$p_n^*(x)$在$[a,b]$上至少有$n+2$个点

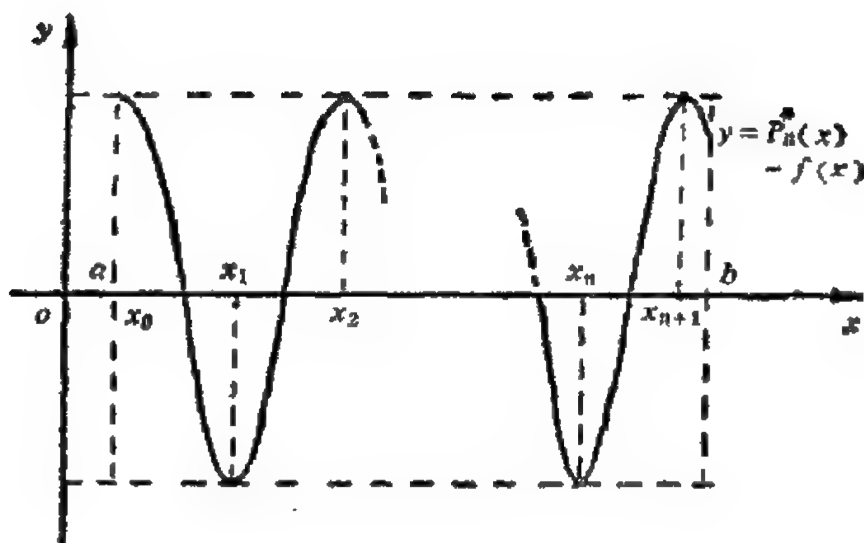
$$a\leqslant x_0<x_1<\cdots<x_{n+1}\leqslant b$$

使

$$p_n^*(x_k)-f(x_k)=(-1)^k\sigma\|p_n^*-f\|_\infty\quad(\sigma=\pm 1)$$

这个定理表明，最佳一致逼近多项式$p_n^*(x)$的特性，即$p_n^*(x)$逼近$f(x)$的误差分布是均匀的如图（3.2.4）所示。

3.2.4 **图**



Чебышев 定理给出了 $p_n^*(x)$ 的充分条件，它是计算 $p_n^*(x)$ 的理论依据，但实际上当 $n\geqslant 2$ 时，求出最佳一致逼近多项式 $p_n^*(x)$ 仍然很困难，这里只对 $n=1$ 的情形给出具体算法。

假定 $f(\cdot)$ 在 $[a,\ b]$ 上二阶导数 $f''(\cdot)$ 连续，且 $f''(\cdot)$ 不变号，设 $p_1^*(x)=a_0+a_1x$，由定理（3.2.3）可知，在区间 $[a,\ b]$ 上至少有 3 个点 $a\leqslant x_0<x_1\leqslant x_2\leqslant b$，使

3.2.5 $$p_1^*(x_k)-f(x_k)=(-1)^k\sigma\|\ p_1^*-f\ \|_\infty \qquad (k=0,\ 1,\ 2)$$

由于 $f''(x)$ 在 $[a,\ b]$ 上不变号，故 $f'(x)$ 在 $[a,\ b]$ 单调，所以，$[p_1^*(x)-f(x)]'=a_1-f'(x)=0$ 只能有一个解，记作 x_1，即 $a_1=f'(x_1)$；另两点必在区间端点，即 $x_0=a$，$x_2=b$，于是由（3.2.5）得

$$p_1^*(a)-f(a)=-[p_1^*(x_1)-f(x_1)]=p_1^*(b)-f(b)$$

由此可得

3.2.6 $$\begin{cases} a_1=\dfrac{f(b)-f(a)}{b-a}=f'(x_1) \\ a_0=\dfrac{f(a)+f(x_1)}{2}-a_1\dfrac{a+x_1}{2} \end{cases}$$

3.2.7 **例** 求$f(x)=e^x$在[-1,1]上的一次最佳一致逼近$p_1^*(x)\in H_1$（如图 3.2.8所示）。

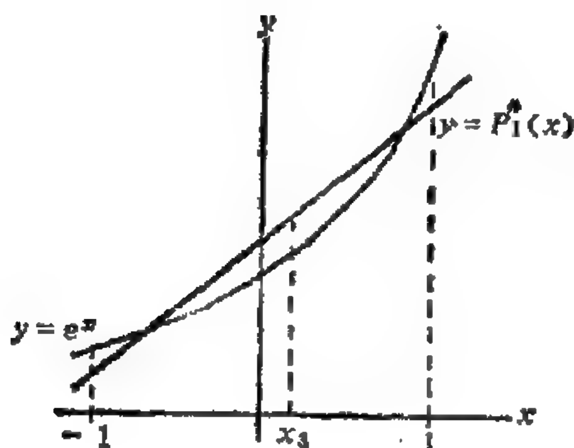
解 令$p_1^*(x)=a_0+a_1x$，由(3.2.6) 得

$$\begin{cases} a_1=\dfrac{e^1-e^{-1}}{2}\approx 1.1752 \\ f'(x_1)=e^{x_1}=a_1\approx 1.1752\rightarrow x_1=\ln a_1\approx 0.1614 \\ a_0=\dfrac{e^{-1}+a_1}{2}+\dfrac{a_1}{2}(1-x_1)\approx 1.2643 \end{cases}$$

于是得到最佳一致线性逼近

$p_1^*(x)=1.2643+1.1752x$，$\Delta(f, p_1^*)\approx 0.2788$

3.2.8 **图**



3.3

3.3 最小平方逼近

因为最佳一致逼近计算困难，所以在实际计算时常使用另一种度量逼近误差的标准，记作

3.3.1 $$\|f-r\|_2^2=\int_a^b\left[f(x)-r(x)\right]^2dx$$

这里$f\in C[a, b]$，$r\in H_n$，用它作度量标准的函数逼近称为**平方逼近**(Squares Approximation)。

$$\frac{1}{\sqrt{b-a}}\| f-r \|_2 = \frac{1}{\sqrt{b-a}}\sqrt{\int_b^a\left[f(x)-r(x)\right]^2 dx}$$

称为**均方误差**(Root-Mean-Square Error)。如果在H_n中存在$r_n^*(x)\in H_n$，使

3.3.2 $$\|f-r_n^*\|_2^2=\int_a^b[f(x)-r_n^*(x)]^2dx=\inf_{r_n\in H_n}\|f-r_n\|_2^2$$

则称 $r_n^*(x)$是 $f(x)$在区间$[a, b]$上的**最小平方逼近**(Least Squares Approximation)。

3.3.3 例 对$f(x)=e^x$，$x\in[-1, 1]$，在H_1中求最小平方逼近多项式.

解 令$r_1(x)=a_0+a_1x$，要求

$$\|f-r_1\|_2^2=\int_{-1}^1(e^x-a_0-a_1x)^2dx\equiv F(a_0, a_1)$$

最小．也就是要求a_0^*及a_1^*，使$F(a_0^*, a_1^*)\leqslant F(a_0, a_1)$，这就是二元函数求极值问题.由极值必要条件

$$\frac{\partial F}{\partial a_0}=0,\quad \frac{\partial F}{\partial a_1}=0$$

可得

$$\frac{\partial F}{\partial a_0}=2\int_{-1}^1[e^x-a_0-a_1x](-1)dx=0$$

$$\frac{\partial F}{\partial a_1}=2\int_{-1}^1[e^x-a_0-a_1x](-x)dx=0$$

由此方程组可解得

$$a_0=\frac{1}{2}\int_{-1}^1e^xdx=\frac{1}{2}(e-e^{-1})\doteq 1.1752$$

$$a_1=\frac{3}{2}\int_{-1}^{1}xe^x dx=3e^{-1}\doteq 1.1036$$

于是在H_1中的最小平方逼近多项式

$r_1^*(x)=1.1752+1.1036x$

直接计算可求得

$\|e^x-r_1^*\|_\infty\approx 0.44$

用 $r_1^*(x)$逼近 e^x 的图可见（3.3.5)(a)。

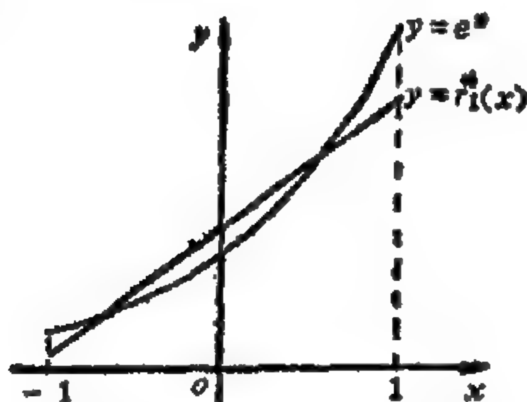
进而可用同样方法计算 e^x在[-1，1]上的三次最小平方逼近

3.3.4 $r_3^*(x)=0.996294+0.997955x+0.536722x^2+0.176139x^3$

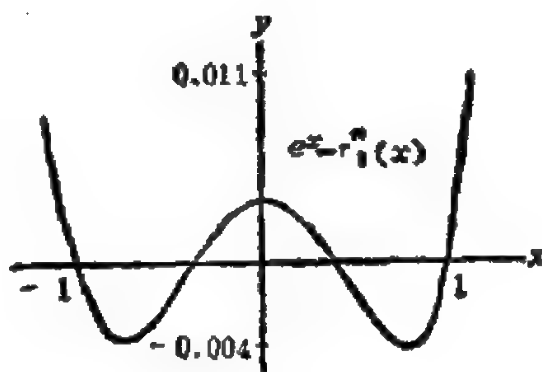
$\|e^x-r_3^*\|_\infty\approx 0.0112$

其误差图形可见（3.3.5)(b)。

3.3.5 图



(a)



(b)

更带普遍性的是带权最小平方逼近。

3.3.6 **定义** 非负函数$\rho(\cdot)$在区间[a，b]（有限或无限）上满足条件：

1° $\int_a^b |x|^n\rho(x)dx$，对所有 $n\geq 0$ 可积。

2° 对非负连续函数 f，若$\int_a^b f(x)\rho(x)dx=0$，则在

$[a, b]$上 $f(\cdot)=0$，称ρ为区间 $[a, b]$ 上的**权函数**(Weight Function).

对给定的 $f\in C[a, b]$，若存在 $p_n^*\in H_n$，使

$$\|f-p^*_n\|_2^2=\int_a^b \rho(x)\left[f(x)-p^*_n(x)\right]^2 dx \tag{3.3.7}$$

$$=\inf_{p\in H_n}\|f-p\|_2^2$$

则称 $p_n^*(\cdot)$ 是 $f(\cdot)$ 在$[a, b]$上带权 $\rho(\cdot)$的**最小平方逼近多项式**. 这里 $\rho(\cdot)$是$[a,b]$上的权函数，当 $\rho=1$时就是(3.3.2)给出的定义，常见的权函数还有

$$\rho(x)=\frac{1}{\sqrt{1-x^2}}\quad(-1\leqslant x\leqslant 1)$$

$$\rho(x)=\sqrt{1-x^2}\quad(-1\leqslant x\leqslant 1)$$

$$\rho(x)=e^{-x}\quad(0\leqslant x<+\infty)$$

$$\rho(x)=e^{-x^2}\quad(-\infty<x<+\infty)$$

为求出 f在$[a, b]$上的最小平方逼近，记

$$p(x)=a_0+a_1x+\cdots+a_nx^n$$

$$F(a_0,\cdots,a_n)=\int_a^b \rho(x)\left[f(x)-\sum_{k=0}^n a_kx^k\right]^2dx \tag{3.3.8}$$

为求系数 $a_k^*(k=0, 1, \cdots, n)$，使

$$F(a_0^*, a_1^*, \cdots, a_n^*)\leqslant F(a_0, a_1, \cdots, a_n)$$

可由多元函数极值必要条件：

$$\frac{\partial F}{\partial a_j}=0\qquad(j=0,1,\cdots,n)$$

得方程组

$$\frac{\partial F}{\partial a_j}=2\int_a^b \rho(x)\left[f(x)-\sum_{k=0}^n a_kx^k\right](-x^j)dx=0$$

$$(j=0,1,\cdots,n)$$

即

3.3.9 $$\sum_{k=0}^{n}\left(\int_a^b \rho(x)x^{k+j}dx\right)a_k=\int_a^b \rho(x)f(x)x^j dx$$

$$(j=0,1,\cdots,n)$$

方程组（3.3.9）称为法方程组（Normal Equations），求（3.3.9）的解可得到 $a_k=a_k^*(k=0,1,\cdots,n)$，当 $\rho(\cdot)\equiv1$、$[a,\ b]=[0,\ 1]$时方程组（3.3.9）为

3.3.10 $$\sum_{k=0}^{n}\frac{a_k}{k+j+1}=\int_0^1 f(x)x^j dx\quad(j=0,1,\cdots,n)$$

a_k的系数矩阵

$$H=\begin{bmatrix}1 & 1/2 & \cdots & \frac{1}{n+1}\\ 1/2 & 1/3 & \cdots & \frac{1}{n+2}\\ \cdots & \cdots & \cdots & \cdots\\ \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n+1}\end{bmatrix}$$

称为Hilbert（希尔伯特）矩阵，显然 $\det(H)\neq0$，方程解 $a_k=a_k^*(k=0,\ 1,\ \cdots,\ n)$ 是唯一的，且关系式$F(a_0^*,\ \cdots,\ a_n^*)\leqslant F(a_0,\ \cdots,\ a_n)$成立，但Hilbert矩阵是一个病态矩阵，当方程（3.3.10）系数或右端项有微小变化时，其解有较大误差；当 $n\geqslant4$ 时，用单字长在计算机进行运算，其结果往往不太可靠，因此，这种算法只适用于 $n\leqslant3$ 的情形，一般情况可用（3.5）提供的正交化方法计算。

3.4

3.4 正交多项式

3.4.1 线性无关与正交函数族

3.4.1 **定义** 给定$f,g\in C[a,b]$，ρ是$[a,b]$上的权函数，称

$$(f,g)=\int_a^b \rho(x)f(x)g(x)dx$$

为函数f与g在$[a,b]$上的**内积**(Inner Product)。

3.4.2 内积有下列简单性质：

1° $(f,g)=(g,f)$；

2° $(af,g)=a(f,g)$, $a\in R^1$；

3° $(f_1+f_2,g)=(f_1,g)+(f_2,g)$；

4° 当$f\neq 0$时，$(f,f)>0$。

利用内积可定义f的Euclid（欧几里得）**范数**(Norm)

3.4.3 $\|f\|_2=\sqrt{(f,f)}$

它满足范数的三条基本性质：

1° 当$f\neq 0$时，$\|f\|_2>0$，当且只当$f=0$时 $\|f\|_2=0$；

2° $\|af\|_2=|a|\ \|f\|_2$, $a\in R^1$；

3° $\|f+g\|_2\leqslant\|f\|_2+\|g\|_2$ （三角不等式）

此外，还有下列结论：

1° $|(f,g)|\leqslant\|f\|_2\ \|g\|_2$，称为 Cauchy-Schwartz（柯西-施瓦兹）不等式；

2° $\|f+g\|_2^2+\|f-g\|_2^2=2\|f\|_2^2+2\|g\|_2^2$，此即平行四边形定律。

3.4.4 **定义** 若内积

$$(f,g)=\int_a^b\rho(x)f(x)g(x)dx=0$$

则称f与g在区间$[a,b]$上带权$\rho(x)$**正交**(Orthogonal)。若函数族$\varphi_0,\varphi_1,\cdots,\varphi_n,\cdots$满足

3.4.5 $$(\varphi_i,\varphi_j)=\int_a^b\rho(x)\varphi_i(x)\varphi_j(x)dx=\begin{cases}0 & (i\neq j)\\ A_j>0 & (i=j)\end{cases}$$

则称$\{\varphi_k\}$是$[a,b]$上带权$\rho(x)$的**正交函数族**(Orthogonal Systems of Functions)。

3.4.6 **例** 三角函数族

1, $\cos x$, $\sin x$, $\cos 2x$, $\sin 2x$, $\cdots$

在区间$[-\pi,\pi]$上正交。因为

$$\begin{cases} \int_{-\pi}^{\pi} \begin{Bmatrix} \cos kx \\ \sin kx \end{Bmatrix} \begin{Bmatrix} \cos jx \\ \sin jx \end{Bmatrix} dx = 0 \quad (k \neq j) \\ \int_{-\pi}^{\pi} \cos kx \sin kx dx = 0 \\ \int_{-\pi}^{\pi} \sin^2 kx dx = \int_{-\pi}^{\pi} \cos^2 kx dx = \pi \end{cases}$$

$(k=1,2,\cdots)$

3.4.7 定义 若φ_0，$\varphi_1,\cdots,\varphi_n$在$[a, b]$上连续，如果

$a_0\varphi_0(x)+\cdots+a_n\varphi_n(x)=0$

当且只当$a_0=\cdots=a_n=0$才成立，就称φ_0，$\varphi_1,\cdots$，φ_n在$[a, b]$上**线性无关**。若函数族$\{\varphi_k, k=0, 1, \cdots\}$中任何有限个$\varphi_k$线性无关，则称其为**线性无关函数族**。例如，函数族1，x，$\cdots$，x^n，$\cdots$就是$[a, b]$上的线性无关函数族。

3.4.8 定理 φ_0，φ_1，$\cdots$，φ_n在$[a, b]$上线性无关的充分必要条件是

$$G_n=G(\varphi_0,\cdots,\varphi_n)=\det\begin{pmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1)\cdots & (\varphi_0,\varphi_n)\\ \vdots & \vdots & \vdots \\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1)\cdots & (\varphi_n,\varphi_n)\end{pmatrix}\neq 0$$

G_n称为Gram（**格拉姆**）**行列式**。

任何线性无关的函数族$\{\varphi_k(x), k=0, 1, \cdots\}$均可通过逐步正交化过程构成正交函数族$\{\psi_k(x), k=0,1,\cdots\}$，其中$\psi_k$是$\varphi_0$，$\cdots$，$\varphi_k$的线性组合。如由单项式$\{1, x, \cdots, x^k, \cdots\}$构成的正交多项式可表示为

3.4.9 $\psi_k(x)=a_kx^k+\cdots a_1x+a_0 \qquad (a_k\neq 0)$

它满足条件

3.4.10 $(\psi_k,\psi_j)=\int_a^b \rho(x)\psi_k(x)\psi_j(x)dx=\begin{cases}0 & (j\neq k)\\ A_k>0 & (j=k)\end{cases}$

其中$\rho(x)$为区间$[a, b]$上的权函数。当$\rho(x)$及$[a, b]$取得不同，便可得到不同的正交多项式。

3.4.2 Legendre多项式

当权函数$\rho(x)\equiv 1$,区间为$[-1,1]$时,由$\{1,x,\cdots,x^n,\cdots\}$正交化得到的正交多项式称为 Legendre（勒让德）多项式，可表示为

3.4.11 $$P_0(x)=1,\quad P_n(x)=\frac{1}{2^n n!}\frac{d^n}{dx^n}[(x^2-1)^n]\quad (n=1,2\cdots)$$

由于$(x^2-1)^n$是$2n$次多项式，求n阶导数得

$$P_n(x)=\frac{1}{2^n n!}(2n)\cdots(n+1)x^n+a_{n-1}x^{n-1}+\cdots+a_0$$

其最高项系数$a_n=\dfrac{(2n)!}{2^n(n!)^2}\neq 0$。最高项系数为1的Legendre多项式可表示为

$$\widetilde{P}_n(x)=\frac{n!}{(2n)!}\frac{d^n}{dx^n}[(x^2-1)^n]$$

Legendre多项式有以下重要性质：

1° 正交性：

3.4.12 $$(P_n,P_m)=\int_{-1}^{1}P_n(x)P_m(x)dx=\begin{cases}0 & (m\neq n)\\ \dfrac{2}{2n+1} & (m=n)\end{cases}$$

2° 奇偶性：

3.4.13 $$P_n(-x)=(-1)^n P_n(x)$$

即n为奇数时$P_n(x)$为奇函数，n为偶数时$P_n(x)$为偶函数。

3° 递推关系：

3.4.14 $$(n+1)P_{n+1}(x)=(2n+1)xP_n(x)-nP_{n-1}(x)\quad (n\geqslant 1)$$

由$P_0(x)=1$，$P_1(x)=x$，利用（3.4.14）就可推出

$P_2(x)=(3x^2-1)/2$

$P_3(x)=(5x^3-3x)/2$

$P_4(x)=(35x^4-30x^2+3)/8$

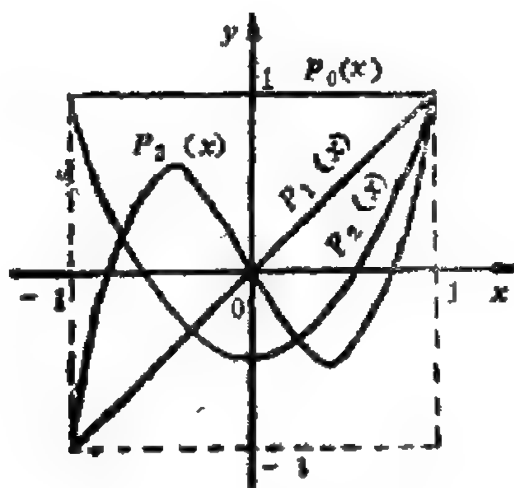
$P_5(x)=(63x^5-70x^3+15x)/8$

$P_6(x)=(231x^6-315x^4+105x^2-5)/16$

…………

图（3.4.15）给出了前面四个Legendre多项式的图形。

3.4.15 图



4° $P_n(x)$在区间[-1，1]内有n个不同的实零点。

5° 在所有最高项系数为1的n次多项式中，$\widetilde{P}_n(x)$在区间[-1，1]上与零的平方误差最小。

3.4.3 Чебышев多项式

当权函数$\rho(x)=\dfrac{1}{\sqrt{1-x^2}}$，区间为[-1，1]时，得到的正交多项式就是Чебышев多项式，它可表示为

3.4.16 $T_n(x)=\cos(n\arccos x)\qquad(|x|\leqslant1)$

令$x=\cos\theta$，则$T_n(x)=\cos n\theta$，$0\leqslant\theta\leqslant\pi$，由三角公式

$T_{n\pm1}(x)=\cos(n\pm1)\theta=\cos(n\theta)\cos\theta\mp\sin(n\theta)\sin\theta$

可得递推关系：

3.4.17 $T_{n+1}(x)=2xT_n(x)-T_{n-1}(x) \quad (n\geqslant 1)$

由（3.4.16）可直接得

$T_0(x)=1$，$T_1(x)=x$

由（3.4.17）可推出

$T_2(x)=2x^2-1$

$T_3(x)=4x^3-3x$

$T_4(x)=8x^4-8x^2+1$

$T_5(x)=16x^5-20x^3+5x$

$T_6(x)=32x^6-48x^4+18x^2-1$

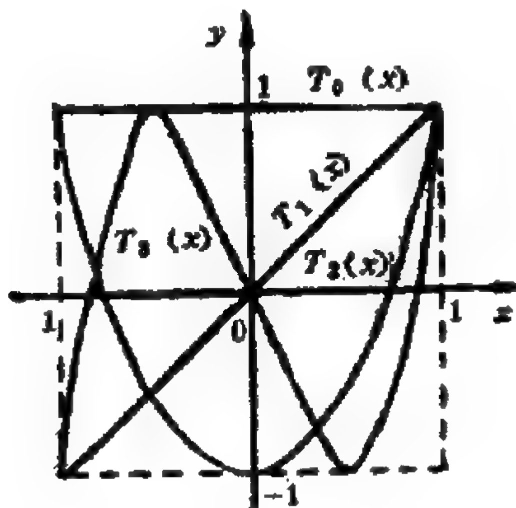
$T_7(x)=64x^7-112x^5+56x^3-7x$

$T_8(x)=128x^8-256x^6+160x^4-32x^2+1$

……

由（3.4.16）可推得$T_n(x)$的最高项系数是2^{n-1}.$T_n(x)$ 前四个的图形见图（3.4.18）

3.4.18 图



Чебышев多项式有以下重要性质：

1° 正交性：$\{T_n(x)\}$在$[-1,1]$上带权$\rho(x)=\dfrac{1}{\sqrt{1-x^2}}$

正交，即

$$\int_{-1}^{1}\frac{T_n(x)T_m(x)}{\sqrt{1-x^2}}dx=\begin{cases}0 & (n\neq m)\\ \dfrac{\pi}{2} & (n=m\neq 0)\\ \pi & (n=m=0)\end{cases}$$

2° 极性：在所有最高项系数为1的多项式中，Чебышев多项式$\widetilde{T}_n(x)=\dfrac{1}{2^{n-1}}T_n(x)$在区间$[-1,1]$上与零偏差最小，且其偏差为$\dfrac{1}{2^{n-1}}$，即

$$\frac{1}{2^{n-1}}=\max_{-1\leqslant x\leqslant 1}\left|\widetilde{T}_n(x)\right|\leqslant\max_{-1\leqslant x\leqslant 1}\left|x^n+P_{n-1}(x)\right|$$

$$(P_{n-1}\in H_{n-1})$$

3° 奇偶性：当n为奇数时$T_n(x)$为奇函数，当n为偶数时$T_n(x)$为偶函数，从而有

$T_n(-x)=(-1)^nT_n(x)$

4° $T_n(x)$在区间$[-1,1]$上有n个实零点

$$x_k=\cos\frac{2k-1}{2n}\pi\quad(k=1,2,\cdots,n)$$

有$n+1$个轮流为"+"、"-"的偏差点

$$y_k=\cos\frac{k\pi}{n}\quad(k=0,1,\cdots,n)$$

5° x^n可用$T_0,T_1,\cdots,T_n$的线性组合表示成（3.4.19）.

3.4.19 表

$1=T_0$

$x=T_1$

$x^2=\dfrac{1}{2}(T_0+T_2)$

$$x^3=\frac{1}{4}(3T_1+T_3)$$

$$x^4=\frac{1}{8}(3T_0+4T_2+T_4)$$

$$x^5=\frac{1}{16}(10T_1+5T_3+T_5)$$

$$x^6=\frac{1}{32}(10T_0+15T_2+6T_4+T_6)$$

$$x^7=\frac{1}{64}(35T_1+21T_3+7T_5+T_7)$$

$$x^8=\frac{1}{128}(35T_0+56T_2+28T_4+8T_6+T_8)$$

3.4.4 其他常用的正交多项式

除上面两类重要的正交多项式外，常用的正交多项式还有以下三种。

1° 第二类Чебышев多项式

在区间[-1，1]上取权函数$\rho(x)=\sqrt{1-x^2}$得到的正交多项式就是第二类Чебышев多项式，其表达式为

3.4.20 $$U_n(x)=\frac{\sin[(n+1)\arccos x]}{\sqrt{1-x^2}},\qquad |x|\leqslant 1$$

令$x=\cos\theta$可得

$$\int_{-1}^{1}U_n(x)U_m(x)\sqrt{1-x^2}\,dx=\begin{cases}0 & (m\neq n)\\ \dfrac{\pi}{2} & (m=n)\end{cases}$$

由（3.4.19）可得递推关系

$U_0(x)=1,\qquad U_1(x)=2x$

$U_{n+1}(x)=2xU_n(x)-U_{n-1}(x)\quad (n=1,2,\cdots)$

2° Laguerre（拉盖尔）多项式

在区间$[0,\ \infty)$上，取权函数$\rho(x)=e^{-x}$得到的正交多项式，就是Laguerre（拉盖尔）多项式，其表达式为

3.4.21 $$L_n(x)=e^x\frac{d^n}{dx^n}(x^ne^{-x})$$

它满足关系

$$\int_0^{\infty}e^{-x}L_n(x)L_m(x)dx=\begin{cases}0 & (m\neq n)\\ (n!)^2 & (m=n)\end{cases}$$

其递推关系为：

$L_0(x)=1,\qquad L_1(x)=1-x$

$L_{n+1}(x)=(1+2n-x)L_n(x)-n^2L_{n-1}(x)\quad(n=1,2,\cdots)$

3° Hermite（埃尔米特）多项式

在区间$(-\infty,\infty)$上，取权函数$\rho(x)=e^{-x^2}$得到的正交多项式就是Hermite（埃尔米特）多项式，其表达式为

3.4.22 $$H_n(x)=(-1)^ne^{x^2}\frac{d^n}{dx^n}(e^{-x^2})$$

它满足正交性条件

$$\int_{-\infty}^{+\infty}e^{-x^2}H_m(x)H_n(x)dx=\begin{cases}0 & (m\neq n)\\ 2^nn!\sqrt{\pi} & (m=n)\end{cases}$$

并有递推关系

$H_0(x)=1,\qquad H_1(x)=2x$

$H_{n+1}(x)=2xH_n(x)-2nH_{n-1}(x)\quad(n=1,2,\cdots)$

3.5

3.5 用正交多项式作最小平方逼近

以单项式$\{1,x,\cdots,x^n\}$为基求最小平方逼近多项式时，其法方程是病态方程组，当n较大时误差很大，这时应采用正交多项式$\{\psi_0(x),\ \cdots,\ \psi_n(x)\}$做基，用广义多项式

$r(x)=a_0\psi_0(x)+\cdots+a_n\psi_n(x)$

作最小平方逼近，由于$\psi_k(x)\in H_k$，故$r(x)\in H_n$，于是在H_n中求$f(\cdot)\in C[a,\ b]$的最小平方逼近，即求

$$\| f-r\|_2^2 =\int_a^b \rho(x)[f(x)-\sum_{j=0}^{n} a_j\psi_j(x)]^2 dx$$

$$=G(a_0,a_1,\cdots,a_n)$$

的最小值，根据（3.3）可知，a_0，a_1，…，a_n应满足方程

3.5.1 $\dfrac{\partial G}{\partial a_k}=\sum_{j=0}^{n}(\psi_j,\psi_k)a_k-(f,\psi_k)=0 \quad (k=0,1,\cdots,n)$

由于$\{\psi_k(x)\}$是正交的，所以由（3.5.1）可得解

$$a_k=\frac{(f,\psi_k)}{(\psi_k,\psi_k)} \qquad (k=0,1,\cdots,n)$$

于是$f(x)$的最小平方逼近为

3.5.2 $r_n^*(x)=\sum_{k=0}^{n} a_k^*\psi_k(x)$

其中

$$a_k^*=\frac{(f,\psi_k)}{(\psi_k,\psi_k)} \qquad (k=0,1,\cdots,n)$$

最小平方误差为

3.5.3 $\| f-r_n^*\|_2^2=\| f\|_2^2 -\sum_{k=0}^{n} a_k^*(f,\psi_k)$

$f(\cdot)$在$[a,\ b]$上按$\{\psi_k(x)\}$展开为

3.5.4 $f(x)\sim \sum_{k=0}^{\infty} a_k^*\psi_k(x)$

此式右端称为$f(x)$的**广义Fourier级数**，a_k^*称为**广义Fourier系数**.级数的部分和$r_n^*(x)$就是$f(x)$在区间$[a,\ b]$上的最小平方逼近.

3.5.1 用Legendre多项式作平方逼近

在有限区间$[a,b]$上求函数$f(x)\in C[a,b]$的最小平方逼近，当$\rho(x)\equiv1$时，只要将$f(\cdot)$按 Legendre（勒让德）多项式展

开即可，但当$[a, b]\neq[-1, 1]$时，应做变换。令

$$x=\frac{b-a}{2}t+\frac{b+a}{2}$$

则$-1\leqslant t\leqslant 1$，于是$f(x)$可变换为

$$F(t)=f\left(\frac{b-a}{2}t+\frac{b+a}{2}\right) \qquad (-1\leqslant t\leqslant 1)$$

对$F(t)$用$[-1, 1]$上的Legendre多项式展开部分和即得$f(x)$在$[a, b]$上的最小平方逼近，因此，只要考虑$f(x)\in C[-1, 1]$，按Legendre多项式$\{P_0, P_1, \cdots, P_n, \cdots\}$展开，由（3.5.4）得

$$f(x)\sim\sum_{k=0}^{\infty}c_k^* P_k(x)$$

其中c_k^*可利用（3.4.12）算得

$$c_k^*=\frac{(f, P_k)}{(P_k, P_k)}=\frac{2k+1}{2}(f, P_k)$$

于是f在$[-1, 1]$上的最小平方逼近为

3.5.5 $$S_n^*(x)=\sum_{k=0}^{n}\frac{2k+1}{2}(f, P_k)P_k(x)$$

这里

$$P_k(x)=\frac{1}{2^k k!}\frac{d^k}{dx^k}[(x^2-1)^k] \qquad (k=1,2,\cdots,n)$$

$$P_0(x)=1$$

由（3.5.3）可得到

$$\|f-S_n^*\|_2^2=\|f\|_2^2-\sum_{k=0}^{n}\frac{2k+1}{2}[(f, P_k)]^2$$

用（3.5.5）求f的最小平方逼近不用求解（3.3.9）的方程组，从而避免了解法方程出现的病态问题。

3.5.6 **例** 求$f(x)=e^x$在[-1，1]上3次最小平方逼近。

解 由（3.5.4）计算(f, P_n) $(n=0, 1, 2, 3)$

$$
\begin{cases}
(f,P_0)=\int_{-1}^{1} e^x dx \approx 2.350388 \\
(f,P_1)=\int_{-1}^{1} xe^x dx \approx 0.735759 \\
(f,P_2)=\int_{-1}^{1} \left(\frac{3}{2}x^2-\frac{1}{2}\right)e^x dx \approx 0.143124 \\
(f,P_3)=\int_{-1}^{1} \left(\frac{5}{2}x^3-\frac{3}{2}x\right)e^x dx \approx 0.201302
\end{cases}
$$

由此可得

$$S_3^*(x)=0.996294+0.997955x+0.536722x^2+0.176139x^3$$

$$\| e^x - S_3^*(x) \|_2^2 \leqslant 0.0084, \qquad \| e^x - S_3^* \|_\infty \leqslant 0.0112$$

这里得到的$S_3^*(x)$就是（3.3.4）的结果。

3.5.2 截断Чебышев级数

如果$f\in C[a, b]$，当正交多项式取Чебышев多项式$\{T_k(x)\}$时，广义Fourier级数

3.5.7 $$\sum_{k=0}^{\infty} c_k^* T_k(x)$$

称为函数f在[-1，1]上的Чебышев级数，其中系数由公式（3.5.2）得

$$
\begin{cases}
c_0^*=\dfrac{(T_0,f)}{(T_0,T_0)}=\dfrac{1}{\pi}(T_0,f)=\dfrac{1}{\pi}\displaystyle\int_{-1}^{1}\frac{f(x)}{\sqrt{1-x^2}}dx \\
c_k^*=\dfrac{(T_k,f)}{(T_k,T_k)}=\dfrac{2}{\pi}(T_k,f)=\dfrac{2}{\pi}\displaystyle\int_{-1}^{1}\frac{T_k(x)f(x)}{\sqrt{1-x^2}}dx
\end{cases}
$$

$$(k=1,2,\cdots)$$

$T_k(x)=\cos(k\arccos x)$为Чебышев多项式.

若令$x=\cos\theta$，$0\leqslant\theta\leqslant\pi$，则（3.5.7)就是$f(\cos\theta)$的Fourier级数，其中

$$c_0^*=\frac{1}{\pi}\int_0^\pi f(\cos\theta)d\theta,\qquad c_k^*=\frac{2}{\pi}\int_0^\pi f(\cos\theta)\cos k\theta d\theta$$

$$(k=1,2,\cdots)$$

根据Fourier级数理论可知，如果$f''(x)$在区间$[-1,1]$上分段连续，则Чебышев级数（3.5.7）在区间$[-1,1]$上一致收敛于$f(x)$。

$$f(x)=\sum_{k=0}^{\infty}c_k^*T_k(x)$$

取它的n部分和

3.5.8 $$c_n^*(x)=\sum_{k=0}^{n}c_k^*T_k(x)$$

则

$$f(x)-c_n^*(x)\approx c_{n+1}^*T_{n+1}(x)$$

由于$T_{n+1}(x)$有$n+2$个轮流为"+"、"−"的偏差点，所以$f(x)-c_n^*(x)$近似地有$n+2$个偏差点，故由Чебышев定理（3.2.3），$c_n^*(x)$可做为$f(x)$的近似最佳一致逼近多项式。事实上，许多函数的截断Чебышев级数（3.5.8）都很近似最佳一致逼近多项式，对具有高阶导数的函数f,其误差$\|f-c_n^*\|_\infty$一般不超过最小偏差E_n的百分之五,最佳一致逼近多项式$p_n^*(x)$难于计算，因此,通常都用$c_n^*(x)$做为近似最佳一致逼近多项式。

3.5.9 例 求$f(x)=e^x$在区间$[-1,1]$上的截断Чебышев级数。

解 由公式（3.5.8），系数为

$$c_0^*=\frac{1}{\pi}\int_{-1}^{1}\frac{e^x}{\sqrt{1-x^2}}dx=\frac{1}{\pi}\int_0^\pi e^{\cos\theta}d\theta$$

$$c_k^* = \frac{2}{\pi}\int_{-1}^{1} f(x)\frac{T_k(x)}{\sqrt{1-x^2}}dx$$

$$= \frac{2}{\pi}\int_0^{\pi} e^{\cos\theta}\cos(k\theta)d\theta \qquad (k=1,2,\cdots)$$

用数值积分方法（见第4章）可算出

$c_0^*=1.26606588,\qquad c_1^*=1.13031821$

$c_2^*=0.27149534,\qquad c_3^*=0.04433685$

$c_4^*=0.00547424,\qquad c_5^*=0.00054293$

由（3.5.8）及$T_k(x)$的表达式，可得

$c_1^*(x)=1.266+1.130x$

$c_3^*(x)=0.994571+0.997308x+0.542991x^2+0.177347x^3$

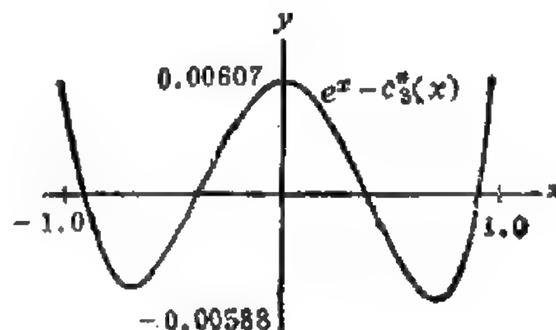
用$c_1^*(x)$及$c_3^*(x)$逼近e^x的误差为

$\| e^x-c_1^*(x)\|_\infty\approx 0.32$

$\| e^x-c_3^*(x)\|_\infty\approx 0.00607$

它与e^x在[-1，1]上的三次最佳一致逼近偏差 0.00553 相差不多.图（3.5.10）给出了误差函数$y=e^x-c_3^*(x)$的图形.它与图（3.7.3）给出的最佳逼近误差分布很相似.这说明用截断的Чебышев级数$c_n^*(x)$，可作为最佳一致逼近的很好近似.

3.5.10 图



3.6

3.6 近似最佳一致逼近

由于函数的最佳一致逼近多项式计算困难,因此,实际计算

往往只求近似的最佳一致逼近，主要工具是利用Чебышев多项式的良好性质，其中以（3.5）给出的截断Чебышев级数 $c_n^*(x)$ 为最好的近似算法，另外，还有下面两种近似方法。

3.6.1 Taylor级数项数的节约

由(3.1.7)可知，函数Taylor展开容易计算，但误差分布极不均匀。为了改善函数逼近的误差分布，可利用 $T_n(x)$ 均匀分布的性质，对Taylor部分和 $P_n(x)$ 进行改造，设

$$\|f(x)-p_n(x)\|_\infty \leqslant \varepsilon_n \ll \varepsilon \tag{3.6.1}$$

其中 $p_n(x)$ 由（3.1.5）给出，$\varepsilon>0$ 是要求逼近的精度。由于

$$\widetilde{T}_n(x)=\frac{1}{2^{n-1}}T_n(x)=x^n+Q(x) \tag{3.6.2}$$

其中 $Q(x)$ 为次数小于 n 的多项式。现设

$$p_n(x)=a_0+a_1x+\cdots+a_nx^n$$

由（3.6.2）可得

$$p_n(x)$$

$$=a_0+a_1x+\cdots+a_{n-1}x^{n-1}+a_n\left[-Q(x)+\frac{1}{2^{n-1}}T_n(x)\right]$$

令

$$M_{n,n-1}(x)=a_0+a_1x+\cdots+a_{n-1}x^{n-1}-a_nQ(x)$$

$$\varepsilon_{n-1}=\left|\frac{a_n}{2^{n-1}}\right|$$

若

$$\varepsilon_n+\varepsilon_{n-1}\leqslant\varepsilon$$

则

$$\|f(x)-M_{n,n-1}\|_\infty\leqslant\|f-p_n\|_\infty+\|p_n-M_{n,n-1}\|$$

$$\leqslant\varepsilon_n+\varepsilon_{n-1}\leqslant\varepsilon$$

这时用 $M_{n,n-1}(x)$ 作为 $f(x)$ 新的逼近多项式，次数已降低一次，再由 $M_{n,n-1}(x)$ 出发，用同样方法，若 $\varepsilon_n+\varepsilon_{n-1}+\varepsilon_{n-2}\leqslant\varepsilon$，则 $M_{n,n-1}(x)$ 可再降低一次，一直做到次数不能再降低为止。

此时得到的逼近多项式记作$M_{n,m}(x)$，$m<n$.

3.6.3 **例** $f(x)=e^x$，$|x|\leqslant 1$，其五阶Taylor展开为

$$p_5(x)=1+x+\frac{1}{2}x^2+\frac{1}{6}x^3+\frac{1}{24}x^4+\frac{1}{120}x^5$$

$$\| f-p_5 \|_\infty \leqslant \frac{1}{6!}e\approx 0.00377$$

由于

$$x^5=\frac{1}{16}T_5(x)+\frac{5}{4}x^3-\frac{5}{16}x$$

$$x^4=\frac{1}{8}T_4(x)+x^2-\frac{1}{8}$$

$$p_5(x)=M_{5,3}(x)+\frac{1}{192}T_4(x)+\frac{1}{1920}T_5(x)$$

其中

$$M_{5,3}(x)=0.994792+0.997396x+0.541667x^2+0.177083x^3$$

$$\| e^x-M_{5,3} \|_\infty \leqslant 0.00377+\frac{1}{192}+\frac{1}{1920}\approx 0.0095$$

进一步若由$p_6(x)$出发，可求得

3.6.4 $$M_{6,3}(x)=0.994575+0.997396x+0.542969x^2+0.177083x^3$$

$$\| e^x-M_{6,3} \|_\infty \leqslant 0.00651$$

这结果已接近$p_3^*(x)$结果，如从$p_7(x)$或$p_8(x)$出发求$M_{7,3}(x)$或$M_{8,3}(x)$，其结果将会更接近$p_3^*(x)$，故可做为求最佳一致逼近的近似算法。

3.6.2 Чебышев多项式零点插值

利用插值多项式$L_n(x)$逼近$f(x)$，若插值点选择合适，可使误差减小。设$f\in C[-1,1]$，选$T_n(x)$的零点$x_k=\cos\frac{2k-1}{2n}\pi$

$(k=1, \cdots, n)$为插值点，造插值多项式$L_{n-1}(x)$，由于

$$\omega_n(x)=(x-x_1)\cdots(x-x_n)=\frac{1}{2^{n-1}}T_n(x)$$

故插值误差

$$\|f-L_{n-1}\|_\infty \leqslant \frac{M_n}{n!}\max_{-1\leqslant x\leqslant 1}|\omega_n(x)|=\frac{M_n}{n!}\frac{1}{2^{n-1}}$$

其中

$$M_n=\max_{-1\leqslant x\leqslant 1}|f^{(n)}(x)|$$

由Чебышев多项式性质2°可知，$\frac{1}{2^{n-1}}T_n(x)$ 在$[-1, 1]$上最高项系数为1的 n 次多项式中与零偏差最小，即 $\max\limits_{-1\leqslant x\leqslant 1}|\omega_n(x)|=\min$，这说明选$T_n(x)$的零点做插值点在固定$M_n$后其误差最小，因此，可利用这种插值多项式$L_{n-1}(x)$ 做为近似最佳一致逼近。当$f(x)\in C[a, b]$时，只要通过变换

$$x=\frac{b-a}{2}t+\frac{b+a}{2}$$

即可在$t\in[-1, 1]$上应用$T_n(t)$ 的零点做插值点，求得所要求的插值多项式。

3.6.5 例 求$f(x)=e^x$的插值多项式。

解 在$[-1,1]$求$L_3(x)$，用$T_4(x)$的零点$x_k=\cos\frac{2k-1}{8}\pi$ $(k=1,2,3,4)$做插值点。

$$x_1=\cos\frac{1}{8}\pi\approx 0.9238795, \quad x_2=\cos\frac{3}{8}\pi\approx 0.3826834$$

$$x_3=\cos\frac{5}{8}\pi\approx -0.3826834, \quad x_4=\cos\frac{7}{8}\pi\approx 0.9238795$$

造插值多项式可得

$$L_3(x)=0.994584+0.998967x+0.542900x^2+0.175176x^3$$

$$\|e^x-L_3\|_\infty\leqslant 0.00666$$

3.7

3.7 Remes 算法

Чебышев 定理 (3.2.3) 从理论上给出了求最佳一致逼近多项式 $P_n^*(x)$的方法，设$f\in C[a,b]$，其最佳逼近多项式

$$p_n^*(x)=\sum_{k=0}^{n}a_k^*x^k$$

的 $n+1$ 个系数 $a_k^*(k=0,1,\cdots,n)$，最小偏差 E_n 和$n+2$个偏差点 $a\leqslant x_0^*<x_1^*<\cdots<x_{n+1}^*\leqslant b$，一共为 $2n+4$ 个未知数。由定理 (3.2.3) 应满足方程

3.7.1
$$\begin{cases}\sum_{j=0}^{n}a_j^*x_k^j-f(x_k^*)=(-1)^k\sigma E_n\begin{pmatrix}\sigma=\pm 1\\ k=0,1,\cdots,n+1\end{pmatrix}\\ (x_0^*-a)(x_{n+1}^*-b)[p_n^{*\prime}(x_k^*)-f'(x_k^*)]=0\\ \qquad\qquad (k=0,1,\cdots,n+1)\end{cases}$$

这是一个$2n+4$个未知数的非线性方程组，一般情况是很难求解的，为了求 $p_n^*(x)$，可由(3.7.1)出发，利用Чебышев多项式先求出近似程度好的偏差点，再用逐次逼近方法求出$p_n^*(x)$，这就是Remes(里姆斯) 算法(Algorithms of Remes)。为方便起见，仍假定$f\in C[-1,1]$，算法分三步进行。

1° 给出 $n+2$ 个初始偏差点

$$-1\leqslant x_0<x_1<\cdots<x_{n+1}\leqslant 1$$

通常，可用 $T_{n+1}(x)$ 的偏差点 $x_k=\cos\left(\frac{k\pi}{n+1}\right)(k=0,1,\cdots,n+1)$解 $n+2$ 个未知数 $a_0,\cdots,a_n$ 及 E 的线性方程组

$$a_0+a_1x_k+\cdots+a_nx_k^n-f(x_k)=(-1)^kE$$

$$(k=0,1,\cdots,n+1)$$

从而得到逼近多项式 $p_n(x)=\sum_{k=0}^{n}a_kx^k$ 及 E。

2° 求 $n+2$ 个新的偏差点

$-1\leqslant z_0<z_1<\cdots<z_{n+1}\leqslant 1$

要求 $p_n(z_k)-f(z_k)$正负交错，且

$p_n'(z_k)-f'(z_k)=0 \quad (k=1, \cdots, n)$

也可包括 z_0 及 z_{n+1}。如上式只有 n 个点成立，则可取 $z_0=-1$，$z_{n+1}=1$. 在某些点 z_k 上，满足

$\|f-p_n\|_\infty=|f(z_k)-p_n(z_k)|$

3° 根据定理（3.2.3）和偏差点$\{z_k\}$的性质，有

$$m=\min_k|f(z_k)-p_n(z_k)|\leqslant\|f-p_n\|_\infty$$
$$\leqslant M=\max_k|f(z_k)-p_n(z_k)|$$

如 M/m 充分靠近1，则 $p_n(x)$为所求，记作 $p_n^*(x)$。例如，当 $M/m\leqslant 1.05$ 时则得 $p_n^*(x)$；否则，用点 $\{z_k\}$代替$\{x_k\}$转回1°继续迭代。

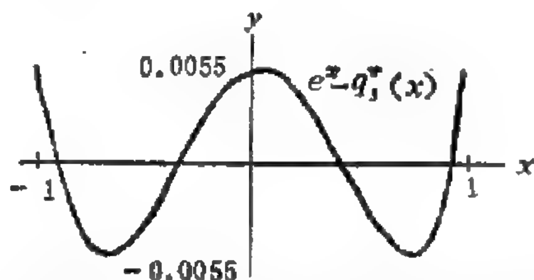
根据 Remes 算法，可求得$f(x)=e^x$ 在〔-1，1〕上的最佳一致逼近

3.7.2 $$p_3^*(x)=0.994579+0.995668x+0.542973x^2+0.179533x^3$$

$\|f-p_n^*\|_\infty=0.00553$

$p_n^*(x)$逼近$f(x)$的误差分布见图（3.7.3）。

3.7.3 图 e^x的三次最佳逼近误差



下面，以e^x在〔-1,1〕上的二次逼近多项式为例，比较各

种逼近的误差，列表如下：

3.7.4 表

逼 近 方 法	误差 $\|f-p\|$
Taylor多项式$p_3(x)$	0.0516
级数节约项数$M_{5,3}(x)$	0.0095
级数节约项数$M_{6,3}(x)$	0.00651
Legendre最小平方逼近$S_3^*(x)$	0.0112
Чебышев最小平方逼近$C_3^*(x)$	0.00607
Чебышев零点插值$L_3(x)$	0.00666
最佳一致逼近$p_3^*(x)$	0.00553

从表中可看到，利用Чебышев多项式得到的一些逼近多项式已经是最佳一致逼近的较好近似，它们计算较简单，而Remes算法计算复杂，通常不大使用。

3.8 曲线拟合的最小二乘法

3.8.1 基本原理

在科学实验或统计方法研究中，往往要从一组实验数据(x_i, y_i) $(i=1, \cdots, m)$中寻找自变量x和因变量y之间的一个函数关系式$y=f(x)$，从图形上看就是由给定的m个点求曲线拟合的问题。由于科学实验得到的数据往往带有观测误差，如果要求曲线$y=f(x)$必须通过给定的点(x_i, y_i)，不但会把观测误差保留下来，而且$y=f(x)$的曲线也不一定表示实验数据的客观规律。因此，对这类问题不能使用第2章介绍的插值方法，而应采用最小二乘法，所谓曲线拟合的最小二乘法，就是由给定数据(x_i, y_i) $(i=1, \cdots, m)$，确定拟合曲线类型$y=P(x, a_0, \cdots, a_n)$，然后根据在给定点误差平方和

3.8.1 $\sum_{i=1}^{m}[P(x_i,a_0,\cdots,a_n)-y_i]^2=\min$

的原则定出参数 $a_k(k=0,\ \cdots,\ n)$，从而得到所求的拟合曲线方程$y=P^*(x)$。

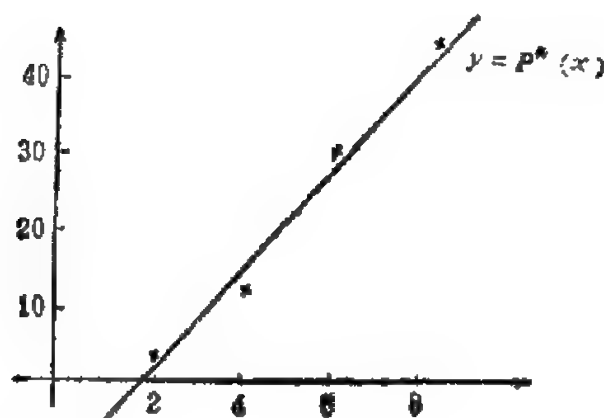
例 已给一组实验数据如下：

i	1	2	3	4
x_i	2	4	6	8
y_i	2	11	28	40

将它标在坐标纸上，如图（3.9.2）所示．容易看出这些点在一条直线附近，因此，可以选

$P(x)=a_0+a_1x$

3.8.2 图



由（3.8.1）令

$$F(a_0,a_1)=\sum_{i=1}^{4}[(a_0+a_1x_i)-y_i]^2$$

为求二元函数 F 的最小点，由极值必要条件可得

$$\begin{cases} \dfrac{\partial F}{\partial a_0} = \sum_{i=1}^{4} 2(a_0 + a_1 x_i - y_i) = 0 \\ \dfrac{\partial F}{\partial a_1} = \sum_{i=1}^{4} 2x_i(a_0 + a_1 x_i - y_i) = 0 \end{cases}$$

化简

$$\begin{cases} 4a_0 + \left(\sum_{i=1}^{4} x_i\right) a_1 = \sum_{i=1}^{4} y_i \\ \left(\sum_{i=1}^{4} x_i\right) a_1 + \left(\sum_{i=1}^{4} x_i^2\right) a_1 = \sum_{i=1}^{4} x_i y_i \end{cases}$$

将已知数据代入后得

$$\begin{cases} 4a_0 + 20a_1 = 81 \\ 20a_0 + 120a_1 = 536 \end{cases}$$

解得

$a_0 = -12.5$, $\quad a_1 = 6.55$

于是得到拟合曲线方程

$y = P^*(x) = 6.55x - 12.5$

3.8.2 一般的最小二乘逼近

设给定数据 (x_i, y_i) $(i=1, \cdots, m)$，假定拟合曲线方程为 $y = P(x, a_0, \cdots, a_n)$，令

$$F(a_0, \cdots, a_n) = \sum_{i=1}^{m} \omega_i [P(x_i, a_0, \cdots a_n) - y_i]^2$$

$$(n < m+1)$$

$\omega_i > 0$ $(i=1, \cdots, m)$称为点 (x_i, y_i) 的权系数，最小二乘法就是求参量 $a_i (i=1, \cdots, m)$使

$F(a_0, \cdots, a_n) = \min$

即求 $a_i = a_i^*$ 使 $F(a_0^*, a_1^*, \cdots, a_n^*) \leqslant F(a_0, \cdots, a_n)$。

由多元函数极值必要条件可得

$$3.8.3 \quad \frac{\partial F}{\partial a_k}=0 \qquad (k=0,1,\cdots,n)$$

当 F 为 a_l 的非线性函数时称为非线性最小二乘问题，这时(3.8.3) 为非线性方程组（求解方法见第 9 章）。如果 F 为a_l 的线性函数，则称为线性最小二乘问题，此时可令

$P(x)=a_0\varphi_0(x)+a_1\varphi_1(x)+\cdots+a_n\varphi_n(x)$

其中 φ_0, φ_1, …, φ_n 是线性无关的．此时

$$3.8.4 \quad F(a_0,\cdots,a_n)=\sum_{i=0}^{m}\omega_i\left[\sum_{k=0}^{m}a_k\varphi_k(x_i)-y_i\right]^2$$

由（3.8.3）可得

$$3.8.5 \quad \frac{\partial F}{\partial a_k}=2\sum_{i=1}^{m}\omega_i\left[\sum_{k=0}^{n}a_k\varphi_k(x_i)-y_i\right]\varphi_k(x_i)=0$$

$$(k=0,1,\cdots,n)$$

若记 $y_i=f(x_i)$,

$$(\varphi_k,\varphi_j)=\sum_{i=1}^{m}\omega_i\varphi_k(x_i)\varphi_j(x_i), \quad (f,\varphi_j)=\sum_{i=1}^{m}\omega_i f(x_i)\varphi_j(x_i)$$

则（3.8.5）式可改写为

$$3.8.6 \quad \sum_{k=0}^{n}(\varphi_k,\varphi_j)a_k=(f,\varphi_j) \qquad (j=0,1,\cdots,n)$$

这个方程组也称**法方程**。它是关于 a_0, …, a_n 的 $n+1$ 阶线性方程组，其系数矩阵为

$$\Phi=\begin{pmatrix}(\varphi_0,\varphi_0) & (\varphi_0,\varphi_1)\cdots(\varphi_0,\varphi_n)\\ \cdots & \cdots \qquad \cdots\\ (\varphi_n,\varphi_0) & (\varphi_n,\varphi_1)\cdots(\varphi_n,\varphi_n)\end{pmatrix}$$

由于$\{\varphi_0, \varphi_1, \cdots, \varphi_n\}$线性无关，故$\det\Phi\neq0$，方程组（3.8.6）存在唯一解，可用消去法求得$a_k=a_k^*$　$(k=0,1\cdots,n)$，从而得到离散数据(x_i, y_i)的最小二乘拟合曲线

3.8.7 $y=P_n^*(x)=a_0^*\varphi_0(x)+a_1^*\varphi_1(x)+\cdots+a_n^*\varphi_n(x)$

可以证明，这样得到的 $P_n^*(x)$确实使（3.8.4）取得极小，故 $P_n^*(x)$即为所求最小二乘解。若取 $\varphi_k(x)=x^k$，$(k=0,1,\cdots,n)$这时得到的最小二乘解为

$P_n^*(x)=a_0^*+a_1^*x+\cdots+a_n^*x^n$

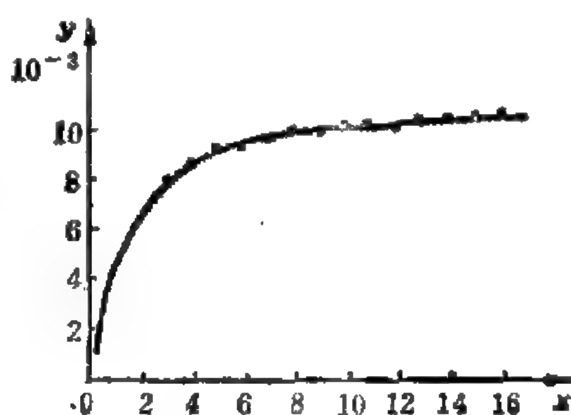
与连续情形最小平方逼近相似，相应的系数矩阵 Φ 也是病态的。因此，当 $n\geqslant4$ 时，用这种方法计算误差较大，通常要用正交化方法计算，见（3.9）。对于实际问题中遇到的大量属于 $n\leqslant3$ 的情形，均可用本节介绍的算法。

3.8.8 例 在某化学反应里，根据实验，生成物的浓度与时间关系如下表，求浓度 y 与时间 t 的拟合曲线。

时间t(分)	1	2	3	4	5	6	7	8
浓度$y\times10^{-3}$	4.00	6.40	8.00	8.80	9.22	9.50	9.70	9.80
时间t(分)	9	10	11	12	13	14	15	16
浓度$y\times10^{-3}$	10.0	10.20	10.32	10.42	10.50	10.55	10.59	10.60

解 先将实验数据表点在坐标纸上，见图（3.8.9）。

3.8.9 图



然后，确定经验公式的函数类型。从图可以看到浓度 y 开始增加快而后逐渐减弱，到一定时间后基本稳定，$t\to\infty$ 时 $y\to$常数，有一水平渐近线；反应

未开始时 $t=0, y=0$。根据这些分析可设想下列两种曲线：

双曲线型

$$1/y=a+\frac{b}{t}$$

即 $y=\frac{t}{at+b}$，此时 $t=0$，$y=0$；$t\to\infty$，$y\to\frac{1}{a}$。

指数型

$y=ae^{b/t}$

当 $b<0$ 时，有

$t\to 0 \quad y\to 0;\qquad t\to+\infty,\quad y\to a.$

第三步，求最小二乘法的解.为确定系数a，b，可以分别两种类型计算.对双曲线型，可做变换$\overline{y}=\frac{1}{y}$，$x=\frac{1}{t}$，则得线性模型

$\overline{y}=a+bx=p(x)$

拟合曲线数据由(t_i, y_i)变换为$(x_i, \overline{y}_i)(i=1, \cdots, 16)$.以下计算与例（3.8.2）相似，其法方程为

$$\begin{cases}16a+3.38073b=1.8372\times 10^3\\ 3.38073a+1.58435b=0.52886\times 10^3\end{cases}$$

解得

$a=80.6621,\qquad b=161.6822$

从而得到

$$y=\frac{t}{80.6621t+161.6822}=F^{(1)}(t) \tag{3.8.10}$$

对指数型

$$y=ae^{b/t} \tag{3.8.11}$$

两边取对数，得

$\ln y=\ln a+\frac{b}{t}$

做变换

$\hat{y}=\ln y$, $A=\ln a$, $x=\frac{1}{t}$

由(t_i,y_i)计算$(x_i, \hat{y}_i)$，数据$(x_i,\hat{y}_i)$的拟合曲线为线性模型

$\hat{y}=A+bx=\phi(x)$

用线性拟合最小二乘法可求得

$A=-4.48072$，$b=-1.0567$，$a=e^A=11.3253\times10^{-3}$

由（3.8.11）可得

$$y=11.3253\times10^{-3}e^{-\frac{1.0567}{t}}=F^{(2)}(t) \tag{3.8.12}$$

第四步，根据求得的函数$F^{(1)}$及$F^{(2)}$计算，得

$\delta_i^{(1)}=y_i-F^{(1)}(t_i)$及$\delta_i^{(2)}=y_i-F^{(2)}(t_i)$　　$(i=1,\cdots,16)$

$\max\limits_i\left|\delta_i^{(1)}\right|=0.568\times10^{-3}$，$\max\limits_i\left|\delta_i^{(2)}\right|=0.277\times10^{-3}$.

可见，$\|\delta^{(2)}\|_\infty$比$\|\delta^{(1)}\|_\infty$小。因此，选择$y=F^{(2)}(t)$作拟合曲线较好。

在用最小二乘法做曲线拟合时，数学模型选择要通过实际计算才能确定，目前有很多计算机配有自动选择数学模型的软件，它通过大量计算可选得误差较小的模型。另外，只要通过变换能化为线性模型的问题均可用本节的算法计算。

3.8.3　多元最小二乘拟合

最小二乘法的有关概念与结论，可以推广到多元函数。例如，假定已知一组正数$\omega_i(i=1, \cdots, m)$和多元函数

$y=f(x_1,\cdots,x_l)$

的一组测量数据$(x_{1i},\cdots,x_{li},y_i)(i=1, \cdots, m)$，要求函数

$$p(x_1,\cdots,x_l)=\sum_{k=0}^{n}a_k\varphi_k(x_1,\cdots,x_l)\quad(n<m+1)$$

使得

$$F(a_0,\cdots,a_n)=\sum_{i=1}^{m}\omega_i[y_i-p(x_{1i},\cdots,x_{li})]^2$$

最小，这也是线性最小二乘拟合问题。

$\{\omega_i\}$为权系数，其中系数 a_0，a_1，$\cdots$，a_n 同样满足法方程组(3.8.6)，只是这里

$$(\varphi_j,\varphi_k)=\sum_{i=1}^{m}\omega_i\varphi_j(x_{1i},\cdots,x_{li})\varphi_k(x_{1i},\cdots,x_{li})$$

求解法方程组得到的$y=p(x_1,\cdots,x_l)$,称为函数$y=f(x_1,\cdots,x_l)$ 的最小二乘拟合函数。

3.9

3.9 用正交多项式作最小二乘拟合

用多项式作最小二乘拟合时，其法方程是病态的，为了避免求病态方程，通常采用正交化方法，对给定点集$\{x_i\}$ $(i=1,\cdots,m)$ 及权系数$\{\omega_i\}$ $(i=1,\cdots,m)$，如果函数系$\varphi_0(x)$，$\cdots$，$\varphi_n(x)$满足

3.9.1 $$(\varphi_j,\varphi_k)=\sum_{i=1}^{m}\omega_i\varphi_j(x_i)\varphi_k(x_i)=\begin{cases}0 & (j\neq k)\\ A_k>0 & (j=k)\end{cases}$$

则称$\{\varphi_j\}$关于点集$\{x_i\}$带权$\{\omega_i\}$正交。此时方程 (3.8.6)解为

3.9.2 $$a_k=a_k^*=\frac{(f,\varphi_k)}{(\varphi_k,\varphi_k)}=\frac{\sum_{i=1}^{m}\omega_i f(x_i)\varphi_k(x_i)}{\sum_{i=1}^{m}\omega_i\varphi_k^2(x_i)}$$

$(k=0,1,\cdots,n)$

且平方误差为

$$3.9.3 \quad \|\delta\|_2^2=\|f\|_2^2-\sum_{k=0}^{n}A_k(a_k^*)^2$$

若已给数据(x_i, f_i)及权ω_i $(i=1,\cdots,m)$，可以造出带权$\{\omega_i\}$正交的多项式$\{p_k\}$，用递推关系表示

$$3.9.4 \quad \begin{cases} p_0(x)=1 \\ p_1(x)=(x-\alpha_1)p_0(x) \\ p_{k+1}(x)=(x-\alpha_{k+1})p_k(x)-\beta_k p_{k-1}(x) \end{cases}$$

$$(k=1,2,\cdots,n-1)$$

这里 p_k 是首项系数为 1 的 k 次多项式，待定系数 α_k 及 β_k 可根据$\{p_k(x)\}$的正交性，即

$$(p_k,p_j)=\sum_{i=1}^{m}\omega_i p_k(x_i)p_j(x_i)=\begin{cases} 0 & (j\neq k) \\ A_k>0 & (j=k) \end{cases}$$

$$(j,k=0,1,\cdots,n)$$

求得

$$3.9.5 \quad \begin{cases} \alpha_{k+1}=\dfrac{\sum_{i=1}^{m}\omega_i x_i p_k^2(x_i)}{\sum_{i=1}^{m}\omega_i p_k^2(x_i)}=\dfrac{(x_i p_k,p_k)}{(p_k,p_k)} \\ \\ \beta_k=\dfrac{\sum_{i=1}^{m}\omega_i p_k^2(x_i)}{\sum_{i=1}^{m}\omega_i p_{k-1}^2(x_i)}=\dfrac{(p_k,p_k)}{(p_{k-1},p_{k-1})} \end{cases} \quad (k=0,1,\cdots,n-1)$$

用点集$\{x_i\}$带权$\{\omega_i\}$正交的多项式$\{p_k(x)\}$为基的广义多项式

$S_n(x)=a_0p_0(x)+a_1p_1(x)+\cdots+a_np_n(x)$

作最小二乘法，由(3.9.2)可得

3.9.6 $a_k = a_k^* = \frac{(f, p_k)}{(p_k, p_k)} = \frac{\sum_{i=1}^{m} \omega_i f_i p_k(x_i)}{\sum_{i=1}^{m} \omega_i p_k^2(x_i)}$

于是

$$S_n^*(x) = a_0^* p_0(x) + a_1^* p_1(x) + \cdots + a_n^* p_n(x)$$

就是所求的最小二乘拟合曲线，其平方误差由（3.9.3）给出。用正交化方法求 $S_n^*(x)$时，只要对 $k=0,1,\cdots,n-1$ 同时用公式（3.9.4）、(3.9.5) 及（3.9.6）计算 $p_k(x)$ 及 a_k^*，其计算方法简单，又不用解方程组，因此是一个较好的算法，一般数学库中有用这个算法编制的标准程序供用户调用。

3.10

3.10 Fourier逼近

当 $f(x)$是定义在$-\infty<x<\infty$的周期函数时，用三角函数逼近$f(x)$比用多项式逼近更为合适。用三角函数作最小平方逼近，亦即函数的 Fourier(傅里叶)逼近及快速Fourier变换 (Fast Fourier Transform)，简称FFT。

3.10.1 最小平方逼近与三角插值

设 $f(x)$是以 2π 为周期的平方可积函数，用**三角多项式** (Trigonometric Polynomials)

$$S_n(x) = a_0 + a_1\cos x + b_1\sin x + \cdots + a_n\cos nx + b_n\sin nx$$

作逼近函数，由例（3.4.6）可知三角函数族

$$1, \cos x, \sin x, \cdots, \cos kx, \sin kx, \cdots$$

是正交函数族，于是$f(x)$在$[0, 2\pi]$上的最小平方三角逼近是

3.10.1 $S_n^*(x)=\frac{1}{2}a_0^*+\sum_{k=1}^{n}(a_k^*\cos kx+b_k^*\sin kx)$

其中

$$a_k^*=\frac{1}{\pi}\int_0^{2\pi}f(x)\cos kx\,dx\qquad(k=0,1,\cdots,n)$$

$$b_k^*=\frac{1}{\pi}\int_0^{2\pi}f(x)\sin kx\,dx\qquad(k=1,2,\cdots,n)$$

称为 Fourier 系数。由$f(x)$的 Fourier 系数得到的级数

3.10.2 $\frac{1}{2}a_0^*+\sum_{k=1}^{\infty}(a_k^*\cos kx+b_k^*\sin kx)$

称为 $f(x)$的 Fourier 级数．只要 $f'(x)$ 在 $[0,\ 2\pi]$ 上分段连续，则级数（3.10.2）一致收敛于 $f(x)$.

与上述情形对应，当 $f(x)$在给定离散点集$\left\{x_i=\frac{2\pi}{n}i\right\}$上的值已知时，可类似定义离散情形的正交性并得到离散 Fourier 系数。对奇数个点的情形，

3.10.3 $x_i=\frac{2\pi i}{2m+1}\qquad(i=0,1,\cdots,2m)$

对 $k,\ j=0,\ 1,\ \cdots,\ m$ 有

3.10.4
$$\begin{cases}\sum_{i=0}^{2m}\sin jx_i\sin kx_i=\begin{cases}0 & (j\neq k,\ j=k=0)\\ \frac{2m+1}{2} & (j=k\neq 0)\end{cases}\\ \sum_{i=0}^{2m}\cos jx_i\cos kx_i=\begin{cases}0 & (j\neq k)\\ \frac{2m+1}{2} & (j=k\neq 0)\\ 2m+1 & (j=k=0)\end{cases}\\ \sum_{i=0}^{2m}\cos jx_i\sin kx_i=0\qquad(\text{对所有}j,k)\end{cases}$$

这就证明了三角函数$\{1, \sin x, \cos x, \cdots, \sin mx, \cos mx\}$关于点集（3.10.3）正交。

假定$f(\cdot)$在点集（3.10.3）上的观测值为$f_i=f(x_i)$，则$f(\cdot)$的最小二乘三角逼近式为

$$S_n(x)=\frac{1}{2}a_0+\sum_{k=1}^{n}(a_k\cos kx+b_k\sin kx), \quad n<m$$

其中

3.10.5 $$\begin{cases} a_k=\dfrac{2}{2m+1}\displaystyle\sum_{i=0}^{2m} f_i\cos\dfrac{2\pi ik}{2m+1} \\ b_k=\dfrac{2}{2m+1}\displaystyle\sum_{i=0}^{2m} f_i\sin\dfrac{2\pi ik}{2m+1} \end{cases} \quad (k=0,1,\cdots n)$$

应用（3.9.3）的结果和正交性（3.10.4）可得

3.10.6 $$\delta_n^2=\sum_{i=0}^{2m}[f_i-S_n(x_i)]^2$$
$$=\sum_{i=0}^{2m}f_i^2-\frac{2m+1}{2}\left[\frac{a_0^2}{2}+\sum_{k=1}^{n}(a_k^2+b_k^2)\right]$$

当$n=m$时，最小二乘逼近就成为三角插值

3.10.7 $$S_m(x)=\frac{1}{2}a_0+\sum_{k=1}^{m}(a_k\cos kx+b_k\sin kx)$$

其中系数a_k，b_k仍由（3.10.5）给出，可证明：

$S_m(x_i)=f_i \qquad (i=0,1,\cdots,2m)$

从而$\delta_m^2=0$，于是由（3.10.6）得

$$\frac{2m+1}{2}\left[\frac{a_0^2}{2}+\sum_{k=1}^{m}(a_k^2+b_k^2)\right]=\sum_{i=0}^{2m}f_i^2$$

3.10.8 例 用下列给出的数据表求$n=1, 2, 3$的Fourier最小二乘逼近。

x_j	0	$2\pi/9$	$4\pi/9$	$2\pi/3$	$8\pi/9$
f_j	3.0004	5.7203	3.1993	−1.0981	−0.8679

x_j	$10\pi/9$	$4\pi/3$	$14\pi/9$	$16\pi/9$
f_j	2.9890	4.0983	1.1477	−0.1882

解 由公式（3.10.5）可求得

$a_0=4.00022$，$a_1=0.99998$，$a_2=0.00011$，$a_3=0.00023$

$b_1=0.00029$，$b_2=2.99997$，$b_3=0.00002$

用公式（3.10.6）可算出

$\delta_0^2=44.99932$，$\delta_1^2=40.49950$，$\delta_2^2=0.00031$，$\delta_3^2=0.00031$

通常可用$\frac{\delta_n^2}{2(m-n)}$的最小者确定 n 的值，在此例中，显然 $n=2$ 给出的最小平方逼近最好，故可得 Fourier 最小二乘逼近为

$$S_2(x)=2.00011+0.99998\cos x+0.00029\sin x$$
$$+0.00011\cos 2x+2.99997\sin 2x$$

事实上，前面给出的数据是由

$$f(x)=2+\cos x+3\sin 2x$$

的摄动得到的，而 $S_2(x)$确实能较好地逼近 $f(x)$。

更一般的情形，假定$f(\cdot)$是以 2π 为周期的复函数，给定在 N 个点$x_j=\frac{2\pi}{N}j(j=0,1,\cdots,N-1)$上的值$f_j=f\left(\frac{2\pi}{N}j\right)$，由于

$$e^{ijx}=\cos(jx)+i\sin(jx)\quad(i=\sqrt{-1},j=0,1,\cdots N-1)$$

故函数族

$1, e^{ix}, \cdots, e^{i(N-1)x}$

在$[0, 2\pi]$上是正交的。函数 e^{ikx}在等距点集$\left\{x_j = \frac{2\pi}{N} j\right\}$上的值 e^{ikx_j} 组成的向量，记作

$$\Phi_k = \left(1, e^{ik\frac{2\pi}{N}}, \cdots, e^{ik\frac{2\pi}{N}(N-1)}\right)^T \quad (k=0,1,\cdots,N-1)$$

N 个向量 ϕ_0, ϕ_1, …, ϕ_{N-1} 也是正交的，即

3.10.9
$$(\phi_k, \phi_l) = \sum_{j=0}^{N-1} e^{ik\frac{2\pi}{N}j} e^{-il\frac{2\pi}{k}j}$$
$$= \sum_{j=0}^{N-1} e^{i(k-l)\frac{2\pi}{N}j} = \begin{cases} 0 & (l \neq k) \\ N & (l = k) \end{cases}$$

因此，$f(x)$在 N 个点 $\{x_j\}$ 上的有限 Fourier 展式为

3.10.10
$$S(x) = \sum_{k=0}^{n-1} c_k e^{ikx} \quad (n \leqslant N)$$

其中

3.10.11
$$c_k = \frac{1}{N} \sum_{j=0}^{N-1} f_j e^{-ikj\frac{2\pi}{N}} \quad (k=0,1,\cdots,N-1)$$

在（3.10.10）中，$n=N$ 时 $S(x)$ 即为 $f(x)$ 在点$\{x_j\}$上的插值函数。此时有

3.10.12
$$f_j = \sum_{k=0}^{N-1} c_k e^{ik\frac{2\pi}{N}j} \quad (j=0,1\cdots,N-1)$$

(3.10.11)称为**离散 Fourier变换**(Discrete Fourier Transform)，简称 DFT；而（3.10.12）则称为反变换。DFT（3.10.11）也是 Fourier 变换

$$H(x) = \int_{-\infty}^{+\infty} f(t) e^{-2\pi i x t} dt$$

离散化的结果，DFT 是用计算机进行 Fourier 分析的主要方法，它在数字讯号处理、全息技术、光谱和声谱分析等很

多领域都有广泛应用。

3.10.2 快速 Fourier 变换(FFT)

无论用公式(3.10.5)计算 Fourier 逼近的系数 a_k 及 b_k，还是用(3.10.11)及(3.10.12)计算 DFT，都可归结为计算

$$c_k=\sum_{j=0}^{N-1} x_j\omega^{kj} \qquad (k=0,1,\cdots,N-1) \tag{3.10.13}$$

其中 $\omega=e^{-i\frac{2\pi}{N}}$ 或 $\omega=e^{i\frac{2\pi}{N}}$，$\{x_j\}$ $(j=0,1,\cdots N-1)$

是已知复数序列。如直接用(3.10.13)计算一个 c_k 要用 N 次复数乘法和 N 次复数加法，称为 N 个操作，计算全部 c_k 共要 N^2 个操作，当 N 很大时运算量也很大，大量数据处理时使用高速计算机有时也无法计算，直到六十年代中期产生了适合计算机使用的快速 Fourier 变换 (FFT) 算法，才大大提高了运算速度。FFT算法的基本思想是尽量减少乘法次数。例如，$ab+ac+ad=a(b+c+d)$，左端三次乘法而右端只用一次乘法，实际上，

$$e^{i\frac{2\pi}{N}kj}=\cos\frac{2\pi}{N}kj+i\sin\frac{2\pi}{N}kj \qquad (k,j=0,1,\cdots,N-1)$$

中只有 N 个不同的值，特别当 $N=2^p$ 时，只有 $\frac{N}{2}$ 个不同的值，因此，在计算 (3.10.13) 时，可合并大量同因子的项，从而减少乘法次数，以 $N=2^3=8$ 为例。由 (3.10.13)，当 $N=8$ 时，得

$$c_k=\sum_{j=0}^{7} x_j\omega^{kj} \qquad (k=0,1,\cdots,7) \tag{3.10.14}$$

将 j，k 用二进制表示为

$j=j_2 2^2+j_1 2^1+j_0 2^0=(j_2 j_1 j_0)$

$k=k_2 2^2+k_1 2^1+k_0 2^0=(k_2 k_1 k_0)$

其中j_r，k_r（$r=0.1$，2）只能取 0 或 1，此时$0\leqslant j$，$k\leqslant 7$

相应记

$c_k=C(k_2 k_1 k_0)$，　$x_j=X(j_2 j_1 j_0)$

于是公式（3.10.14）可表示为

$$C(k_2 k_1 k_0)=\sum_{j_0=0}^{1}\sum_{j_1=0}^{1}\sum_{j_2=0}^{1}x(j_2 j_1 j_0)\omega^{(k_2 k_1 k_0)(j_2 j_1 j_0)}$$

$$=\sum_{j_0=0}^{1}\left\{\sum_{j_1=0}^{1}\left[\sum_{j_2=0}^{1}x(j_2 j_1 j_0)\omega^{k_0(j_2 j_1 j_0)}\right]\omega^{k_1(j_1 j_0 0)}\right\}\omega^{k_2(j_0 00)}$$

若引入记号

$A_0(j_2 j_1 j_0)=x(j_2 j_1 j_0)$

$$\text{3.10.15}\quad\begin{cases}A_1(j_1 j_0 k_0)=\sum_{j_2=0}^{1}A_0(j_2 j_1 j_0)\omega^{k_0(j_2 j_1 j_0)}\\[2ex]A_2(j_0 k_1 k_0)=\sum_{j_1=0}^{1}A_1(j_1 j_0 k_0)\omega^{k_1(j_1 j_0 0)}\\[2ex]A_3(k_2 k_1 k_0)=\sum_{j_0=0}^{1}A_2(j_0 k_1 k_0)\omega^{k_2(j_0 00)}\end{cases}$$

则

$C(k_2 k_1 k_0)=A_3(k_2 k_1 k_0)$

说明计算$c_k=C(k_2 k_1 k_0)$可分为 3 步进行；对一般$N=2^p$的情形，则计算c_k可分为p步进行，注意

$$\omega^{k_0 2^{p-1}}=\omega^{k_0\frac{N}{2}}=(-1)^{k_0}$$

公式（3.10.15）还可化简：

$$A_1(j_1 j_0 k_0)=\sum_{j_2=0}^{1}A_0(j_2 j_1 j_0)\omega^{k_0(j_2 j_1 j_0)}$$

$$=A_0(0j_1j_0)\omega^{k_0(0j_1j_0)}+A_0(1j_1j_0)\omega^{k_02^2}\omega^{k_0(0j_1j_0)}$$
$$=[A_0(0j_1j_0)+(-1)^{k_0}A_0(1j_1j_0)]\omega^{k_0(0j_1j_0)}$$

即

$$\begin{cases}A_1(j_1j_00)=A_0(0j_1j_0)+A_0(1j_1j_0)\\A_1(j_1j_01)=[A_0(0j_1j_0)-A_0(1j_1j_0)]\omega^{(0j_1j_0)}\end{cases}$$

将二进制还原为十进制表示$j=(0j_1j_0)=j_12^1+j_02^0$，即 $j=0$，1，2，3时得

3.10.16 $$\begin{cases}A_1(2j)=A_0(j_1)+A_0(j+2^2)\\A_1(2j+1)=[A_0(j)-A_0(j+2^2)]\omega^j\quad(j=0,1,2,3)\end{cases}$$

同理，（3.10.15）中的 A_2 可简化为：

3.10.17 $$\begin{cases}A_2(j2^2+k)=A_1(2j+k)+A_1(2j+k+2^2)\\A_2(j2^2+k+2)=[A_1(2j+k)-A_1(2j+k+2^2)]\omega^{2j}\end{cases}$$
$$(k,j=0,1)$$

（3.10.15）中的 A_3 可简化为：

3.10.18 $$\begin{cases}A_3(k)=A_2(k)+A_2(k+2^2)\\A_3(k+2^2)=A_2(k)-A_2(k+2^2)\qquad(k=0,1,2,3)\end{cases}$$

根据公式（3.10.16）-（3.10.18），由 $A_0(j)=x(j)(j=0,1,\cdots,7)$出发逐次计算到 $A_3(k)=c_k$，就得到所要的结果，总共只做八次复数乘法就求出全部 $c_k(k=0,\cdots,7)$.

将上面得到的 $N=2^3$ 的计算公式(3.10.16)-(3.10.18)推广到 $N=2^p$ 的一般情形可得FFT的计算公式如下：

3.10.19 $$\begin{cases}A_q(j2^q+k)=A_{q-1}(j2^{q-1}+k)+A_{q-1}(j2^{q-1}+k+2^{p-1})\\A_q(j2^q+k+2^{q-1})=[A_{q-1}(j2^{q-1}+k)-A_{q-1}(j2^{q-1}+k+2^{p-1})]\omega^{j2^{q-1}}\end{cases}$$

$$\begin{pmatrix}q=1,\cdots,p;\ j=0,1,\cdots,2^{p-q}-1\\k=0,1,\cdots,2^{q-1}-1\end{pmatrix}$$

这里 A_q 括号内的数代表序号，亦即在计算机中存放该数的

地址号；一组 A_q 占用 N 个复数单元，计算时只需用两组单元轮流替换，由 $A_0(j)=x(j)(j=0,1,\cdots,N-1)$ 出发，$q=1,\cdots,p$，用公式（3.10.19）计算到 $A_p(k)=c_k$（$k=0,1,\cdots,N-1$）即为所求，在计算机上的计算步骤见算法（3.10.20），计算一组 A_q 共做 $2^{p-q}\cdot2^{q-1}=2^{p-1}=\frac{N}{2}$ 次复数乘法，而最后一步计算 A_p 时，由于 $\omega^{j\cdot2^{p-1}}=(\omega^{\frac{N}{2}})^j=(-1)^j=(-1)^0=1$（注意：当 $q=p$ 时 $2^{p-q}-1=0$，故 $j=0$），因此，当 $q=p$ 时计算 A_p 不用乘法，故共需要计算 $(p-1)\frac{N}{2}$ 次复数乘法，这比直接用公式（3.10.13）计算需 N^2 次复数乘法少的多，其比值为 $\frac{p-1}{2}:N$。如当 $N=2^{10}$ 时为 $4.5:1024\approx1:230$；它比通常的FFT算法用 Np 次复数乘法也快一倍以上。因此，(3.10.19)的计算公式称为改进的FFT算法，其计算程序步骤如下：

3.10.20 **算法**（改进FFT）

1° 给出复数数组 $A_0(N)$，$A_1(N)$ 及 $\omega\left(\frac{N}{2}\right)$，确定 p。将已知数据 $\{x_j\}$ 输入到数组 $A_0(N)$ 的单元中。

2° 计算 $\omega^m=e^{-i\frac{2\pi}{N}m}$（或 $\omega^m=e^{i\frac{2\pi}{N}m}$）($m$ 从0到 $2^{p-1}-1$)，结果存放在 $\omega(m)$ 的相应单元中。

3° j 从 0 到 $(2^{p-q}-1)$，k 从 0 到 $(2^{q-1}-1)$ 按公式(3.10.19)计算 A_q，结果存放在 A_1 的相应单元。即求

$$A_1(j2^q+k)=A_0(j2^{q-1}+k)+A_0(j2^{q-1}+k+2^{p-1})$$

$$A_1(j2^q+k+2^{q-1})=[A_0(j2^{q-1}+k)-A_0(j2^{q-1}+k+2^{p-1})]\,\omega(j2^{q-1})$$

圆括号内的数表示地址号，即数据存放位置。

4° k,j 循环结束，$q+1\to q$，$A_1(N)\to A_0(N)$。若 $q<p$，则

转步骤 3°，否则执行步骤 5°.

5° 此时 $q=p$，$j=0$，k 从 0 到 $(2^{p-1}-1)$，计算

$A_1(k)=A_0(k)+A_0(k+2^{p-1})$

$A_1(k+2^{p-1})=A_0(k)-A_0(k+2^{p-1})$

这里得到的 $A_1(k)$ ($k=0, 1, \cdots, 2^p-1$) 即为所求 c_k.

3.11

3.11 有理逼近与连分式

多项式是函数逼近的一种很好的工具，但用多项式逼近计算函数值并不是最简的，特别当函数具有极点时，用有理函数逼近比用多项式逼近要精确得多. 所谓有理函数是指形如

3.11.1 $$R_{mn}(x)=\frac{P_m(x)}{Q_n(x)}=\frac{a_mx^m+\cdots+a_1x+a_0}{b_nx^n+\cdots+b_1x+b_0}$$

的有理分式，其中 P_m 与 Q_n 分别为 m 次和 n 次多项式. 虽然有理函数计算要用除法运算，但在计算机上除法与乘法的运算时间大体相同，而有理逼近又可减少乘除法运算次数，因此，计算机上使用的函数子程序有很多都用有理逼近. 用 (3.11.1) 的有理逼近其效果粗略地等同于 $m+n$ 次多项式的逼近能力，但其误差通常比多项式逼近小. 连分式在有理逼近与有理分式计算中起重要作用. 如在计算 $R_{m,n}(x)$ 中，化为连分式时只要用 m 或 n 次乘除法运算，比 $m+n$ 次多项式用 $m+n$ 次乘法运算减少了运算量.

展开式

3.11.2 $$b_0+\cfrac{a_1}{b_1+\cfrac{a_2}{b_2+\ddots+\cfrac{a_n}{b_n+\ddots}}}$$

称为连分式 (Continued Fraction)，它可表示为

$$b_0+\frac{a_1}{b_1}\,+\frac{a_2}{b_2}\,+\cdots+\frac{a_n}{b_n}\,+\cdots$$

分式 $\frac{a_n}{b_n}$ 称为连分式(3.11.2)的第 n 节，a_n，b_n 称为连分式 (3.11.2) 第 n 节的两项，a_1，…，a_n，…叫做连分式的部分分子，b_1，…，b_n，…叫做连分式的部分分母。假定所有 b_i 均不为零，有限连分式

$$b_0+\frac{a_1}{b_1}\,+\cdots+\frac{a_n}{b_n}=\frac{P_n}{Q_n}$$

称为连分式 (3.11.2) 的第 n 个渐近分式。相邻三个渐近分式之间有递推关系：

$$\begin{cases}P_n=b_nP_{n-1}+a_nP_{n-2}\\Q_n=b_nQ_{n-1}+a_nQ_{n-2}\end{cases}\tag{3.11.3}$$

事实上，按连分式定义

$$\frac{P_0}{Q_0}=\frac{b_0}{1},\ \frac{P_1}{Q_1}=b_0+\frac{a_1}{b_1}=\frac{b_0b_1+a_1}{b_1}$$

$$\frac{P_2}{Q_2}=b_0+\frac{a_1}{b_1}\,+\frac{a_2}{b_2}=b_0+\frac{a_1b_2}{b_1b_2+a_2}=\frac{b_2P_1+a_2P_0}{b_2Q_1+a_2Q_0}$$

若取 $P_{-1}=1$，$Q_{-1}=0$，则当 $n=1$ 和 $n=2$ 时 (3.11.3) 式成立。假设 (3.11.3) 对 n 成立，要证明它对 $n+1$ 也成立，只要注意由 $\frac{P_n}{Q_n}$ 推导 $\frac{P_{n+1}}{Q_{n+1}}$ 时，用 $b_n+\frac{a_{n+1}}{b_{n+1}}$ 代替 b_n 即可得到

$$\frac{P_{n+1}}{Q_{n+1}}=\frac{b_nP_{n-1}+\frac{a_{n+1}}{b_{n+1}}\ P_{n-1}+a_nP_{n-2}}{b_nQ_{n-1}+\frac{a_{n+1}}{b_{n+1}}Q_{n-1}+a_nQ_{n-2}}$$

$$= \frac{b_{n+1}(b_n P_{n-1} + a_n P_{n-2}) + a_{n+1} P_{n-1}}{b_{n+1}(b_n Q_{n-1} + a_n Q_{n-2}) + a_{n+1} Q_{n-1}}$$

$$= \frac{b_{n+1} P_n + a_{n+1} P_{n-1}}{b_{n+1} Q_n + a_{n+1} Q_{n-1}}$$

从而证明 (3.11.3) 对一切 n 成立。

3.11.4 例 $\ln(1+x)$当$x>-1$时有如下的展开式

$$\ln(1+x) = \frac{x}{1} + \frac{x}{2} + \frac{x}{3} + \frac{2^2 x}{4} + \frac{2^2 x}{5} + \frac{3^2 x}{6} + \frac{3^2 x}{7} + \cdots$$

将它化为有理分式。

解 先计算

$$\frac{P_0}{Q_0} = \frac{x}{1}, \quad \frac{P_1}{Q_1} = \frac{x}{1} + \frac{x}{2} = \frac{2x}{2+x}$$

再由公式 (3.11.3) 求得

$$\frac{P_2}{Q_2} = \frac{3(2x) + x(x)}{3(x+2) + x \cdot 1} = \frac{6x + x^2}{6 + 4x}$$

$$\frac{P_3}{Q_3} = \frac{4(6x + x^2) + 4x(2x)}{4(6 + 4x) + 4x(2 + x)} = \frac{4(6x + 3x^2)}{4(6 + 6x + x^2)}$$

$$\frac{P_4}{Q_4} = \frac{20(6x + 3x^2) + 4x(6x + x^2)}{20(6 + 6x + x^2) + 4x(6 + 4x)} = \frac{4(30x + 21x^2 + x^3)}{4(30 + 36x + 9x^2)}$$

$$\frac{P_5}{Q_5} = \frac{4[6(30x + 21x^2 + x^3) + 9x(6x + 3x^2)]}{4[6(30 + 36x + 9x^2) + 9x(6 + 6x + x^2)]}$$

$$= \frac{12(60x + 60x^2 + 11x^3)}{12(60 + 90x + 36x^2 + 3x^3)}$$

……

注意：利用递推公式 (3.11.3) 计算时分子与分母的公因数是不能随便约去的，但作为最后结果的有理分式可以约去公因数。

利用 $\ln(1+x)$的连分式展开，可获得前 4 个有理逼近：

$$
\begin{cases}
R_{11}(x)=\dfrac{2x}{2+x} \\[2ex]
R_{22}(x)=\dfrac{6x+3x^2}{6+6x+x^2} \\[2ex]
R_{33}(x)=\dfrac{60x+60x^2+11x^3}{60+90x+36x^2+3x^3} \\[2ex]
R_{44}(x)=\dfrac{420x+630x^2+260x^3+25x^4}{420+840x+540x^2+120x^3+6x^4}
\end{cases}
$$

$\ln(1+x)$的 Taylor 展开前 $2n$ 项和

$$
p_{2n}(x)=x-\frac{x^2}{2}+\frac{x^3}{3}-\cdots+(-1)^{2n-1}\frac{x^{2n}}{2n}
$$

$$
(-1<x\leqslant 1)
$$

$R_{nn}(x)$与 $p_{2n}(x)$ 有相同个数的参数，但它们逼近$\ln(1+x)$的精度差别很大，在用 $R_{nn}(1)$与 $p_{2n}(1)$作为 $\ln 2$的近似时，其误差 ε_R 与 ε_p 的大小情况如下表：

n	$R_{nn}(1)$	ε_R	$p_{2n}(1)$	ε_p
1	0.667	0.026	0.50	0.19
2	0.69231	0.00084	0.58	0.11
3	0.693122	0.000025	0.617	0.076
4	0.69314642	0.00000076	0.634	0.058

$\ln 2$ 的精确值为 $\ln 2=0.69314718\cdots$，$R_{44}(1)$与 $p_8(1)$的精度相差十万倍，这说明某些函数用有理逼近是很必要的。

将有理分式（3.11.1）化为连分式，使乘除运算次数减少。对 $m=n$ 的情形，可得

$$
R_{nn}(x)=\frac{a_nx^n+\cdots+a_1x+a_0}{b_nx^n+\cdots+b_1x+b_0}
$$

$$= \frac{a_n}{b_n} + \cfrac{c_1}{x + \cfrac{a_{n-1}^{(1)} x^{n-1} + \cdots}{b_{n-1}^{(1)} x^{n-1} + \cdots}}$$

$$= c_0 + \frac{c_1}{x + R_{n-1,n-1}(x)}$$

这里 c_0，c_1及$R_{n-1,\,n-1}(x)$可通过除法得到，对 $R_{n-1,\,n-1}(x)$再按同样步骤进行，最后可得

3.11.5 $R_{nn}(x) = c_0 + \frac{c_1}{x+B_1} + \frac{c_2}{x+B_2} + \cdots + \frac{c_n}{x+B_n}$

用这公式计算 $R_{nn}(x)$的值只用 n 次除法运算，若直接用有理分式计算则要用 $2n$ 次乘除法运算。

3.11.6 例　将$R_{33}(x) = \frac{x^3+3x^2-7x-28}{x^3-2x^2-2x-3}$化为连分式

$$R_{33}(x) = 1 + \frac{5(x^2-x-5)}{x^3-2x^2-2x-3} = 1 + \cfrac{5}{x-1+\cfrac{2x-8}{x^2-x-5}}$$

$$= 1 + \cfrac{5}{x-1+\cfrac{2}{\cfrac{x^2-x-5}{x-4}}} = 1 + \frac{5}{x-1} + \frac{2}{x+3} + \frac{7}{x-4}$$

如果有理分式（3.11.1）中 $m>n$，令 $m=n+k$，则可将它表成

3.11.7 $R_{mn}(x) = d_0 + \cdots + d_k x^k + \frac{c_1}{x+B_1} + \cdots + \frac{c_n}{x+B_n}$

用（3.11.7）计算 $R_{mn}(x)$的值需用 $n+k=m$ 次乘除法运算。

若 $m<n$，令 $n=m+k$，则

3.11.8 $R_{mn}(x) = \frac{c_0}{d_0+\cdots+d_{k-1}x^{k-1}+x^k} + \frac{c_1}{x+B_1} + \cdots + \frac{c_m}{x+B_m}$

六

用 (3.11.8) 计算 $R_{mn}(x)$的值也需 $k+m=n$ 次乘除法运算。因此，无论 m，n 为何种情形，将 $R_{mn}(x)$化为连分式计算只需 $\max(n, m)$次乘除法运算，比直接计算 $R_{mn}(x)$时用 $n+m$ 次乘除法运算省，比相应的 $m+n$ 次多项式逼近的乘除法运算次数也少。

3.12

3.12 最佳有理逼近

给定函数$f(\cdot)\in C[a, b]$及一对整数 $m\geqslant 0$ 及 $n\geqslant 0$，用有理分式

3.12.1 $$R_{mn}(x)=\frac{P_m(x)}{Q_n(x)}$$

逼近$f(\cdot)$，这里 $P_m(\cdot)\in H_m$，$Q_n(\cdot)\in H_n$，是次数不超过m和 n 的多项式。设$P_m(\cdot)$与$Q(\cdot)$互质，用 $R_n^m[a, b]$ 表示有理函数类

$$R_n^m[a,b]=\left\{\frac{P_m}{Q_n}, P_m\in H_m, Q_n\in H_n\text{在}[a,b]\text{上}Q_n>0\right\}$$

3.12.2 **定义** 量

$$\Delta(R_{mn})=\sup_{a\leqslant x\leqslant b}|f(x)-R_{mn}(x)|$$

称为$f(\cdot)$与 $R_{mn}(\cdot)$的**偏差**。在 $R_n^m[a, b]$中的最小偏差记作

3.12.3 $$\rho_{mn}(f)=\inf_{R_n^m[a,b]}\Delta(R_{mn})=\inf_{R_n^m}\sup_{x\in[a,b]}|f(x)-R_{mn}(x)|$$

称为函数$f(\cdot)$在有理函数类 $R_n^m[a, b]$中的**最佳有理逼近**(Best Rational Approximations)。若存在$R(x)\in R_n^m[a,b]$，使得

$\Delta(R)=\rho_{mn}(f)$

则称 $R(x)$ 为 $f(\cdot)$ 在有理函数类 $R_n^m[a, b]$ 中的最佳逼近有理分式

3.12.4 定理 设 $f(\cdot)\in C[a, b]$，则在有理函数类 $R_n^m[a, b]$ 中至少存在一个最佳逼近有理分式 $R(x)$。

定理表明连续函数 $f(\cdot)$ 在 $R_n^m[a, b]$ 中最佳逼近有理分式的存在性，通常可将 $R(x)$ 写成

3.12.5 $$R(x)=\frac{a_0+a_1x+\cdots+a_{m-\mu}x^{m-\mu}}{b_0+b_1x+\cdots+b_{n-\nu}x^{n-\nu}}=\frac{A(x)}{B(x)}$$

其中 $A(\cdot)$ 与 $B(\cdot)$ 互质，$b_{n-\nu}\neq 0$，$a_{m-\mu}\neq 0$，$0\leqslant\mu\leqslant m$ $0\leqslant\nu\leqslant n$，记 $d=\min(\mu, \nu)$ 称为 $R(\cdot)$ 的亏项数，如果 $d>0$ 则称 $R(\cdot)$ 是退化的。与多项式的最佳一致逼近 Чебышев 定理类似，对最佳逼近有理分式也有相应定理。

3.12.6 定理 设 $f\in C[a, b]$，在形如（3.12.1）的有理函数类 $R_n^m[a, b]$ 中，有理分式（3.12.5）是最佳逼近有理分式的充要条件是：在 $[a, b]$ 上至少有 $N=m+n-d+2$ 个点，使 $f(x)-R(x)$ 以正负交错的符号达到 $\Delta(R)$。其中 d 是 $R(\cdot)$ 的亏项数，满足上述条件的最佳逼近有理分式 $R(x)$ 是唯一确定的（唯一是指化简后为相同有理分式）。

根据这定理，对 $\mu=\nu=0$ 的非退化情形，求形如

3.12.7 $$R(x)=\frac{a_0+a_1x+\cdots+a_mx^m}{1+b_1x+\cdots+b_nx^n}\qquad (b_0=1)$$

的最佳逼近有理分式，可从一个初始近似 $R^{(0)}(x)$ 出发，用逐次逼近的有理分式序列 $R^{(r)}(x)$ 求得 $R(x)$ 的足够精确的近似。具体实现可用 Remes（里姆斯）算法：

1° 给定初始近似 $R^{(0)}(x)$，在 $[a, b]$ 中选取 $N=m+n+2$ 个点组成的点集 $X_1=\{x_i^{(1)}, i=1, \cdots, N\}$，排列为：

$a\leqslant x_1^{(1)}\leqslant x_2^{(1)}\leqslant\cdots\leqslant x_N^{(1)}\leqslant b$

它使 $f(x_i^{(1)})-R^{(0)}(x_i^{(1)})$ 正负交错，这 N 个点一般就是

$f(x)-R^{(0)}(x)$的极值点。

2° 假设已求出点集 $X_r=\{x_i^{(r)},\ i=1,\ \cdots,N\}$，求 r 次近似

$$R^{(r)}(x)=\frac{a_0^{(r)}+a_1^{(r)}x+\cdots+a_m^{(r)}x^m}{1+b_1^{(r)}x+\cdots+b_n^{(r)}x^n}$$

它的系数可通过求解非线性方程组

3.12.8 $f(x_i^{(r)})-R^{(r)}(x_i^{(r)})=(-1)^iE \qquad (i=1,2,\cdots,N)$

得到。

3° 求取得 $\max\limits_{a\leqslant x\leqslant b}|f(x)-R^{(r)}(x)|$ 的点$\tau\in[a,b]$。

4° 用 τ 取代点集X_r中的某一点，使得$f(x)-R^{(r)}(x)$在新求得的点集$X_{r+1}=\{x_i^{(r+1)},\ i=1,\ \cdots,\ N\}$上交错变号，由于$f(x)-R^{(r)}(x)$在点集 X_r上交错变号，而X_{r+1}只改换X_r的一点，当 τ 落在 $x_i^{(r)}$与 $x_{i+1}^{(r)}$之间时，则 τ 必同 $x_i^{(r)}$或$x_{i+1}^{(r)}$的某一点使$f(x)-R^{(r)}(x)$取同号，此时以 τ 代替这个“同号”的点即可。当 τ 在端点时，可根据 $x_1^{(r)}$与 $x_N^{(r)}$上 $f(x)-R^{(r)}(x)$符号情况决定 τ 究竟应取代那一点。总之，使新点集 X_{r+1} 上 $f(x)-R^{(r)}(x)$交错变号是可能的。

5° 以X_{r+1}代替X_r，重复步骤中的2°，3°，4°，直到$R^{(r)}(x)$满足精度要求为止。

在假定初始近似 $R^{(0)}(x)$ 足够好的前提下，Ralston（赖尔斯顿）给出了 Remes 算法的收敛性定理。

3.12.9 **定理** 设 $R^*(x)$为 $f(x)$的形如（3.12.7）的最佳一致逼近有理分式，上述Remes算法的初始近似 $R^{(0)}(x)$于$[a,b]$上恰有 N 个正负相间的极值点，它的第一个极值与$f(x)-R^*(x)$的第一个极值符号相同，设 $X_1=\{x_1^{(1)},\ \cdots,\ x_N^{(1)}\}$是$f(x)-R^{(0)}(x)$的 N 个极值点的横坐标

$a\leqslant x_1^{(1)}<x_2^{(1)}<\cdots<x_N^{(1)}\leqslant b$

$E_{\min}$ 是误差极值的最小值，又设 $x^*=\{x_i^*\}(i=1,\ \cdots,\ N)$

是$f(x)-R^*(x)$的N个极值点（极值大小为$r_{mn}^*=\|f(x)-R^*(x)\|_\infty$）的横坐标

$$a\leqslant x_1^*<x_2^*<\cdots<x_N^*\leqslant b$$

则有$\varepsilon>0$和$\eta>0$存在，使当

$$|x_i^{(1)}-x_i^*|<\varepsilon \qquad (i=1,\cdots,N)$$

$$r_{mn}^*-E_{\min}<\eta$$

时，只要采用恰当的方法求解（3.12.8），则 Remes 算法收敛，即$\lim\limits_{r\to\infty}R^{(r)}(x)=R^*(x)$.

在定理中要求$R^{(0)}(x)$具有N个正负相间的极值，而在实践中并无必要，应用时只要$R^{(0)}(x)$有N个极值就够了．实际上 Remes算法只要求一个点集X_1，通常可利用$[a, b]$上的 Чебышев 多项式$T_{m+n+1}(x)$的$N=m+n+2$个极值点作为X_1。另一种办法是用某种有理逼近(见(3.14)或(3.15))的极值点，当然这个有理逼近要有$m+n+2$个极值点．

求解方程组（3.12.8）通常可用割线法，实践表明是成功的．Fike（费凯）给出另一种迭代法，先将（3.12.8）化为等价形式

3.12.10 $$a_0+a_1x_i^{(1)}+\cdots+a_m(x_i^{(1)})^m+b_1x_i^{(1)}[(-1)^iE-f(x_i^{(1)})]+\cdots+b_n(x_i^{(1)})^n[(-1)^iE-f(x_i^{(1)})]+(-1)^iE=f(x_i^{(1)}) \qquad (i=1,\cdots,N)$$

再把此式改写为

3.12.11 $$a_0+a_1x_i^{(1)}+\cdots+a_m(x_i^{(1)})^m+b_1x_i^{(1)}[(-1)^i\overline{E}-f(x_i^{(1)})]+\cdots+b_n(x_i^{(1)})^n[(-1)^i\overline{E}-f(x_i^{(1)})]+(-1)^iE=f(x_i^{(1)}) \qquad (i=1,\cdots,N)$$

取$\overline{E}=0$，则（3.12.11）是关于a_0，…，a_m，b_1，…，b_n和E的线性方程组，可求出a_0，…，a_m，b_1，…，b_n和$E=E^{(1)}$的解．以新的$E^{(1)}$代替方程（3.12.11）的$\overline{E}$，则又可求出a_0，…，a_m，b_1，…，b_n和$E=E^{(2)}$；又以$E^{(2)}$代替$\overline{E}$，

再解出一组 a_0，…，a_m，b_1，…，b_n 和 $E^{(2)}$；…；Fike 证明了该过程收敛。

3.12.12 例 设 $f(x)=\cos\frac{1}{4}\pi x$ 在[-1，1]上用

$$R(x)=\frac{a_0+a_1x+a_2x^2+a_3x^3+a_4x^4+a_5x^5}{1+b_1x+b_2x^2}$$

作最佳一致逼近。

解 因 $f(x)$ 是偶函数，[-1,1]是以原点为对称的区间，所以 $R^*(x)$ 的分子与分母必只出现偶次幂，故只需考虑有理分式

$$R(x)=\frac{a_0+a_2x^2+a_4x^4}{1+b_2x^2}$$

由 Чебышев 逼近理论，满足定理（3.12.6）的交错点集应有对称的 9 个点，它们是：

$-x_1^*=x_9^*$，$-x_2^*=x_8$，$-x_3^*=x_7^*$，$-x_4^*=x_6^*$，$x_5^*=0$

故在 Remes 算法 1°步中，点集 X_1 仍应保持这一关系，可取

$$x_i^{(1)}=\cos\frac{(9-i)}{8}\pi \qquad (i=5,6,7,8,9)$$

相应（3.12.10）的非线性方程组为

$$a_0+a_2(x_i^{(1)})^2+a_4(x_i^{(1)})^4+b_2(x_i^{(1)})^2((-1)^iE-\cos\frac{\pi}{4}x_i^{(1)})$$
$$-E=f(x_i^{(1)}) \qquad (i=5,6,7,8,9)$$

用 Remes 算法只须迭代两次，即可求得

$a_0^*=1.0000000241$，　　$a_2^*=-0.2874648358$

$a_4^*=0.0093933390$，　　$b_2^*=0.0209610796$

其偏差为 $E=0.241\times10^{-7}$

$$\cos\frac{\pi}{4}x$$

$$\approx\frac{1.0000000241-0.2874648358x^2+0.009393339x^4}{1+0.0209610796x^2}$$

3.13 有理函数插值

3.13.1 有理插值的存在唯一性

假定给定 $m+n+1$ 个互异的点

$x_0, x_1, \cdots, x_{m+n}$

和相应的函数值

$f(x_0), f(x_1), \cdots, f(x_{m+n})$

构造一个有理分式函数

3.13.1 $$R_{mn}(x)=\frac{N_m(x)}{D_n(x)}=\frac{a_mx^m+a_{m-1}x^{m-1}+\cdots+a_1x+a_0}{b_nx^n+b_{n-1}x^{n-1}+\cdots+b_1x+b_0}$$

使之满足插值条件

3.13.2 $$R_{mn}(x_j)=f(x_j)\quad(j=0,1,\cdots,m+n)$$

这种问题就是所谓有理函数插值问题．当 $n=0$ 时，R_{mn} 是一个 m 次多项式，插值问题(3.13.2)的解存在唯一．但当 $n>0$ 时，R_{mn} 是一个真正的有理分式函数，插值问题(3.13.2)的解不是存在唯一的．例如，假定 $m=0$，$f(x_j)=0$，$f(x_k)\neq0$ 这种特例，此时

$$R_{0n}(x)=\frac{a_0}{b_nx^n+\cdots+b_1x+b_0}$$

由 $R_{0n}(x_j)=0$ 可推知 $a_0=0$，但 $a_0=0$ 时 $R_{0n}(x_k)\neq0$ 显然不成立，故这个插值问题无解．这就说明满足插值条件（3.13.2）的有理分式函数的解是否存在是有条件的．但可以证明插值问题（3.13.2）的解如果存在则必唯一．这里唯一的概念是指两个有理分式

$$R_1(x)=\frac{P_1(x)}{Q_1(x)},\qquad R_2(x)=\frac{P_2(x)}{Q_2(x)}$$

满足条件

$$P_1(x)Q_2(x)\equiv P_2(x)Q_1(x)$$

称 $R_1(\cdot)$ 与 $R_2(\cdot)$ 等价，并用记号 $R_1(\cdot)\sim R_2(\cdot)$．

如果存在常数 $a\neq0$，使

$P_2(x)=\alpha P_1(x)$,　$Q_2(x)=\alpha Q_1(x)$

则称$R_1(\cdot)$与$R_2(\cdot)$恒等，记作$R_1(\cdot)\equiv R_2(\cdot)$。两个有理分式$R_1(\cdot)$与$R_2(\cdot)$等价，必须而且只须$R_1(\cdot)$和$R_2(\cdot)$的最简分式$\overline{R_1}(\cdot)$与$\overline{R_2}(\cdot)$恒等。因此，在使用时只要两个有理分式等价，则认为它们是同一有理分式而不加区别。有理函数插值的唯一性即建立在这种意义上。

因此，对有理函数插值来说，关键问题是存在性和具体解法。当有理分式（3.13.1）满足插值条件（3.13.2）时，只要分母$D_n(x_j)\neq 0(j=0, 1, \cdots, m+n)$，就有

3.13.3　$N_m(x_j)-f(x_j)D_n(x_j)=0$　　$(j=0,1\cdots,m+n)$

它是关于系数a_m，…，a_0，b_n，…，b_0的线性方程组。这里未知数约去一个常数后实质上只有$m+n+1$个，与方程个数相同，方程（3.13.3）比非线性方程(3.13.2)容易求解，但它们是否等价是有条件的。

3.13.4　**定理**　若线性方程组（3.13.3）有非平凡解，为使满足插值条件（3.13.2）的最简有理分式$R_{mn}(x)=p_m(x)/q_n(x)$存在，必须且只须（3.13.3）的任一非平凡解$N_m^*(x)$或$D_n^*(x)$在约去一切公共因子后所得的互质多项式$A(x)$，$B(x)$仍然是（3.13.3）的解，即

$A(x_j)-f(x_j)B(x_j)=0$　　$(j=0,1\cdots,m+n)$

这个定理给出了方程（3.13.2）与（3.13.3）等价的充分必要条件，但是所给条件不便于检验。定理（3.13.5）给出的是一个便于应用的条件。

3.13.5　**定理**　设(x_j, y_j)中各$x_j(j=0, 1, \cdots, m+n)$是互异的，$y_j=f(x_j)$，$(j=0, 1, \cdots, m+n)$。为使满足插值条件（3.13.2）的最简有理分式

$$R_{mn}(x)=\frac{N_m(x)}{D_n(x)}$$

存在，必须且只须下述各矩阵

3.13.6
$$A_j=\begin{pmatrix} 1 & x_0 & x_0^2 & \cdots & x_0^{m-1} & y_0 & x_0y_0 & \cdots & x_0^{n-1}y_0 \\ \cdots & \cdots & \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{j-1} & x_{j-1}^2 & \cdots & x_{j-1}^{m-1} & y_{j-1} & x_{j-1}y_{j-1} & \cdots & x_{j-1}^{n-1}y_{j-1} \\ 1 & x_{j+1} & x_{j+1}^2 & \cdots & x_{j+1}^{m-1} & y_{j+1} & x_{j+1}y_{j+1} & \cdots & x_{j+1}^{n-1}y_{j+1} \\ \cdots & \cdots & \cdots & \cdots & & \cdots & \cdots & \cdots & \cdots \\ 1 & x_{m+n} & x_{m+n}^2 & \cdots & x_{m+n}^{m-1} & y_{m+n} & x_{m+n}y_{m+n} & \cdots & x_{m+n}^{n-1}y_{m+n} \end{pmatrix}$$

$(j=0,1\cdots,m+n)$

都是非奇异的。

3.13.7 例 给定$(x_0, y_0)=(0, 1)$，$(x_1, y_1)=(1, 0)$和$(x_2, y_2)=(2, 0)$，用形如

$$R_{11}(x)=\frac{a_0+a_1x}{b_0+b_1x}$$

的有理函数对上述三个型值点插值．由于

$$A_0=\begin{pmatrix} 1 & 0 \\ 1 & 0 \end{pmatrix}$$

是一个奇异阵，由定理（3.13.5）知上述插值问题解不存在。

3.13.2 Thiele 倒差商算法

Thiele（蒂埃勒）根据多项式插值的 Newton 差商插值公式（2.4）的思想，为解决有理函数插值问题（3.13.2）设计了一种连分式插值的 Thiele 方法，其表达式为

3.13.8 $$R(x)=a_0+\frac{x-x_0}{a_1}+\frac{x-x_1}{a_2}+\cdots+\frac{x-x_{m+n-1}}{a_{m+n}}$$

按插值条件（3.13.2），可逐一求出上述连分式的各个系数：

$$a_0=\varphi_0[x_0]=f(x_0)$$

3.13.9 $$a_k=\varphi_k[x_0,x_1,\cdots,x_{k-1},x_k]$$

$$= \frac{x_k - x_{k-1}}{\varphi_{k-1}[x_0, x_1, \cdots, x_{k-2}, x_k] - \varphi_{k-1}[x_0, x_1, \cdots, x_{k-2}, x_{k-1}]}$$

$(k=1, 2\cdots, m+n)$

注意：使用这个公式时要求 $a_k \neq 0$。

Thiele 还给出了一种倒差商格式，所谓$f(x)$的一阶倒差商，二阶倒差商，三阶倒差商，…，是指

$$\rho_1(x, x_0) = \frac{x - x_0}{f(x) - f(x_0)}$$

$$\rho_2(x, x_0, x_1) = \frac{x - x_1}{\rho_1(x, x_0) - \rho_1(x_0, x_1)} + f(x_0)$$

$$\rho_3(x, x_0, x_1, x_2) = \frac{x - x_2}{\rho_2(x, x_0, x_1) - \rho_2(x_0, x_1, x_2)} + \rho_1(x_0, x_1)$$

…………

为了得到统一的递推公式，令

$\rho_{-1} = 0, \quad \rho_0(x) = f(x)$

则对于 $n=1, 2, \cdots$，有如下递推关系式

3.13.10 $\rho_n(x, x_0, \cdots, x_{n-1})$

$$= \frac{x - x_{n-1}}{\rho_{n-1}(x, x_0, \cdots, x_{n-2}) - \rho_{n-1}(x_0, \cdots x_{n-1})} + \rho_{n-2}(x_0, \cdots, x_{n-2})$$

于是可推出

3.13.11 $f(x) = f(x_0) + \dfrac{x - x_0}{\rho_1(x, x_0)}$

$$= f(x_0) + \cfrac{x - x_0}{\rho_1(x_0, x_1) + \cfrac{x - x_1}{\rho_2(x, x_0, x_1) - f(x_0)}}$$

$$= f(x_0) + \cfrac{x - x_0}{\rho_1(x_0, x_1) + \cfrac{x - x_1}{\rho_2(x_0, x_1, x_2) - f(x_0) + \cfrac{x - x_2}{\rho_3(x, x_0, x_1, x_2) - \rho_1(x_0, x_1)}}} = \cdots$$

它可任意延伸下去，若取它的各个渐近分式，即可得到一批插值有理分式函数，如在(3.13.11)中取第 $m+n$ 渐近分式

3.13.12 $$R_{mn}(x)=f(x_0)+\frac{x-x_0}{\rho_1(x_0,x_1)+}\frac{x-x_1}{\rho_2(x_0,x_1,x_2)-f(x_0)}$$

$$+\frac{x-x_2}{\rho_3(x_0,x_1,x_2,x_3)-\rho_1(x_0,x_1)}+\cdots$$

$$+\frac{x-x_{m+n-1}}{\rho_{m+n}(x_0,\cdots,x_{m+n})-\rho_{m+n-2}(x_0,\cdots,x_{m+n-2})}$$

则 $R_{mn}(x)$满足

$R_{mn}(x_j)=f(x_j)\quad(j=0,1,\cdots,m+n)$

(3.13.12)给出的插值连分式相当于（3.13.8）中取

$a_0=f(x_0)$

$a_1=\rho_1(x_0,x_1)$

$a_2=\rho_2(x_0,x_1,x_2)-f(x_0)$

$a_j=\rho_j(x_0,x_1,\cdots,x_j)-\rho_{j-2}(x_0,\cdots,x_{j-2})$

$(j=3,\cdots,m+n)$

3.14

3.14 Padé逼近

Padé（巴德）逼近是以函数的幂级数展开为基础，设 $f(\cdot)$在原点邻域内可展成幂级数

3.14.1 $$f(x)=\sum_{j=0}^{\infty}a_jx^j$$

再设$f(\cdot)$ 的有理逼近为

3.14.2 $$R_{LM}(x)=\frac{p_L(x)}{Q_M(x)}=\frac{\sum\limits_{k=0}^{L}p_kx^k}{1+\sum\limits_{k=1}^{M}q_kx^k}$$

其中$p_L(\cdot)\in H_L$，$Q_M(\cdot)\in H_M$，p_k及 q_k 均为待定系数，可由

下列方程确定

3.14.3 $f(x)-\dfrac{P_L(x)}{Q_M(x)}=0(x^{L+M+1})$

用$Q_M(x)$乘（3.14.3）式，并将$f(x)$用（3.14.1）表示，然后比较两端 x^j同次幂系数，即得关于系数 p_0，…，p_L及q_1，…q_M的线性方程组

3.14.4
$$\begin{cases} a_0 & =p_0 \\ a_1+a_0q_1 & =p_1 \\ a_2+a_1q_1+a_0q_2 & =p_2 \\ \vdots & \vdots \\ a_L+a_{L-1}q_1+\cdots+a_{L-M}q_M & =p_L \\ a_{L+1}+a_Lq_1+\cdots+a_{L-M+1}q_M & =0 \\ \vdots & \vdots \\ a_{L+M}+a_{L+M-1}q_1+\cdots+a_Lq_M & =0 \end{cases}$$

其中已规定 $a_n=0$。当 $n<0$时，这是关于 p_0，…，p_L，q_1，…，q_M的$L+M+1$个未知量的线性方程组，方程个数是$L+M+1$，求解时可由最后的M个方程求出 q_1，…，q_M，把值代入前 $L+1$个方程，即可求得$p_0,p_1,\cdots,p_L$。因此，对任何给定的非负整数 L，M，都可由方程组（3.14.4）求出$f(x)$的有理逼近 $R_{LM}(x)=\dfrac{p_L(x)}{Q_M(x)}$，这里$Q_M(0)=1$。这样的有理分式函数 R_{LM}即为 f 的（L，M)阶Padé逼近，用〔L/M〕表示。

3.14.5 **定理** 对任意形式的幂级数$f(x)$，若其 Padé逼近存在，则必唯一。

根据唯一性定理，当方程组（3.14.4）的解存在时，不论用什么方法求得满足（3.14.4）的解$P_L(x)$与 $Q_M(x)$，由它产生的有理分式$R_{LM}(x)$都是所要求的 Padé 逼近〔L/M〕，通常可用被称作“Padé表”的图表来表示〔L/M〕的位置。

3.14.6 Padé表

L \ M	0	1	2	3	
0	[0/0]	[0/1]	[0/2]	[0/3]	⋯
1	[1/0]	[1/1]	[1/2]	[1/3]	⋯
2	[2/0]	[2/1]	[2/2]	[2/3]	⋯
3	[3/0]	[3/1]	[3/2]	[3/3]	⋯
⋮	⋮	⋮	⋮	⋮	⋱

当 $M=0$ 时，就是表中第一列为 f 的 Taylor 展开。若

$$f(x)=e^x=1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots$$

则有

3.14.7 e^x的Padé 表(见137页)

在表中令 $x=1$，可得 e 的相应Padé 逼近，

[1/1]=3

[2/2]=19/7≈2.71428571

[3/3]=193/71≈2.71830986

[4/4]=2721/1001≈2.71828172

[4/4]的逼近与 e 的真值误差仅在第 7 位小数上相差 1。如用连分式计算只用 4 次乘除法，相当于四阶的多项式逼近，但精度却高得多。大量算例表明，在 $N=L+M$ 为一确定常数时，各种可能的[L/M]Padé逼近中，以 L 和 M 相等或接近相等者为最精确。如，当 $N=2n$ 时，应采用 [n/n]Padé 逼近；当 $N=2n+1$ 时，应采用 [$n+1/n$]或 [$n/n+1$]Padé 逼近。总之，应采用 Padé 逼近表的主对角或主对角线附近的 Padé 逼近。

对于 [n/n] Padé 逼近，满足 (3.14.4) 的方程可改为

L \ M	0	1	2	3	4
0	$\frac{1}{1}$	$\frac{1}{1-x}$	$\frac{2}{2-2x+x^2}$	$\frac{6}{6-6x+3x^2-x^3}$	$\frac{24}{24-24x+12x^2-4x^3+x^4}$
1	$\frac{1+x}{1}$	$\frac{2+x}{2-x}$	$\frac{6+2x}{6-4x+x^2}$	$\frac{24+6x}{24-18x+6x^2-x^3}$	$\frac{120+24x}{120-90x+36x^2-8x^3+x^4}$
2	$\frac{2+2x+x^2}{2}$	$\frac{6+4x+x^2}{6-2x}$	$\frac{12+6x+x^2}{12-6x+x^2}$	$\frac{60+24x+3x^2}{60-36x+9x^2-x^3}$	$\frac{360+120x+12x^2}{360-240x+72x^2-12x^3+x^4}$
3	$\frac{6+6x+3x^2+x^3}{6}$	$\frac{24+18x+16x^2+x^3}{24-6x}$	$\frac{60+36x+9x^2+x^3}{60-24x+3x^2}$	$\frac{120+60x+12x^2+x^3}{120-60x+12x^2-x^3}$	$\frac{840+360x+60x^2+4x^3}{840-480x+120x^2-16x^2+x^4}$
4	$\frac{24+24x+12x^2+4x^3+x^4}{24}$	$\frac{120+96x+36x^2+8x^3+x^4}{120-24x}$	$\frac{360+240x+72x^2+12x^3+x^4}{360-120x+12x^2}$	$\frac{840+480x+120x^2+16x^2+x^4}{840-360x+60x^2-4x^3}$	$\frac{1680+840x+180x^2+20x^3+x^4}{1680-840x+180x^2-20x^3+x^4}$

3.14.8 $p_0=a_0$, $\quad p_j=\sum_{i=0}^{j} q_i a_{j-i} \quad (j=1,2,\cdots,n)$

其中 $q_0=1$，q_j是下列方程的解。

$$
3.14.9 \quad \begin{cases}
a_1q_n+a_2q_{n-1}+\cdots+a_nq_1+a_{n+1}=0 \\
a_2q_n+a_3q_{n-1}+\cdots+a_{n+1}q_1+a_{n+2}=0 \\
\cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \quad \cdots \\
a_nq_n+a_{n+1}q_{n-1}+\cdots+a_{2n-1}q_1+a_{2n}=0
\end{cases}
$$

将它写成矩阵形式为

$A_nq+b=0$

其中

$$
A_n=\begin{pmatrix} a_1 & a_2 & \cdots & a_n \\ a_2 & a_3 & \cdots & a_{n+1} \\ \cdots & \cdots & \cdots & \cdots \\ a_n & a_{n+1} & \cdots & a_{2n-1} \end{pmatrix}, \quad q=\begin{pmatrix} q_n \\ q_{n-1} \\ \vdots \\ q_1 \end{pmatrix}, \quad b=\begin{pmatrix} a_{n+1} \\ a_{n+2} \\ \vdots \\ a_{2n} \end{pmatrix}
$$

A_n是对称阵，若 A_n非奇，则方程（3.14.9）有解，将它的解 q_1，…，q_n代入（3.14.9）则得p_0，p_1，…，p_n。于是可得〔n/n〕Padé逼近$R_{nn}(x)$。

为了估计$f(x)$与$R_{nn}(x)$的误差，由（3.14.3）可得

$$
Q_n(x)\sum_{k=0}^{\infty} a_kx^k-p_n(x)=x^{2n+1}\sum_{k=0}^{\infty} r_kx^k
$$

其中 r_k可通过比较 x 高于 x^{2n+1}次幂的系数得到。

3.14.10 $\quad r_k=\sum_{i=0}^{n} q_i a_{2n+k+1-i} \quad (k=0,1,\cdots)$

通常 $|a_k|$ 是减少的，因此当 $|x|\leqslant 1$ 时，误差

$$
\begin{aligned}
E_n&=|f(x)-R_{nn}(x)| \\
&=\left|\frac{x^{2n+1}}{Q_n(x)}\sum_{k=0}^{\infty} r_kx^k\right| \approx |r_0|\ |x^{2n+1}| \leqslant |r_0|
\end{aligned}
$$

由（3.14.10）可得

$$r_0=a_{n+1}q_n+a_{n+2}q_{n-1}+\cdots+a_{2n}q_1+a_{2n+1}$$

将这个方程与（3.14.9）联立，消去q_1，…，q_n就得到 r_0，或由 Gram 法则得

$$q_0=1=\frac{\det\begin{pmatrix} c_1 & c_2 & \cdots & c_n & 0 \\ c_2 & c_3 & \cdots & c_{n+1} & 0 \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ c_{n+1} & c_{n+2} & \cdots & c_{2n} & r_0 \end{pmatrix}}{\det A_{n+1}}=r_0\frac{\det A_n}{\det A_{n+1}}$$

记$\Delta_n=\det A_n$，由此求得

$$r_0=\frac{\Delta_{n+1}}{\Delta_n}$$

在求Padé逼近$[n/n]$之前，为了先确定 n，可先估计误差

$$E_n\approx|r_0|=\left|\frac{\Delta_{n+1}}{\Delta_n}\right| \qquad (当\ |x|\leqslant 1)$$

当E_n达到精度要求再计算$[n/n]$。

3.14.11 例 $f(x)=\cos x$，令$x^2=z$，考虑 $|x|\leqslant 1$，

$$\cos x=\cos\sqrt{z}=\sum_{k=0}^{\infty}\frac{(-1)^k}{(2k)!}z^k$$

取$L=M=3$，计算Δ_3，Δ_4。

解 因为$a_k=\dfrac{(-1)^k}{(2k)!}$，故

$$r_0=\frac{\Delta_4}{\Delta_3}=\frac{45469}{59\times 660\times 14!}<1.34\times 10^{-11}$$

当 $|x|\leqslant 1$，则 $E_n<1.34\times10^{-11}$。若 $|x|\leqslant\frac{\pi}{3}=1.048$，而 $(1.048)^{14}\leqslant 2$，故 $E_n\leqslant r_0(1.048)^{14}<2.68\times10^{-11}$，仍能达到

10位有效数字. 由 (3.14.9) 可求得

$q_1=0.0294437956$

$q_2=0.000425456685$

$q_3=0.00000326957732$

由 (3.14.8) 可求得

$p_0=1$

$p_1=-0.4705562044$

$p_2=0.0273702256$

$p_3=-0.0003715227$

于是得到

$$\cos x\approx\frac{1+p_1x^2+p_2x^4+p_3x^6}{1+q_1x^2+q_2x^4+q_3x^6}$$

3.15

3.15 Maehly逼近

Padé逼近是以函数的幂级数展开为基础的. 其误差近似为r_0x^{L+M+1}, 当 $|x|\ll 1$ 时误差很小, 但当 $|x|$ 增大时误差迅速增大, 特别当函数的幂级数展开收敛较慢时, 效果较差. Maehly (梅利) 逼近是 Padé 逼近的改进, 它利用$f(\cdot)$的Чебышев展开求函数的有理逼近, 使逼近精度大大提高.

设$f(x)$在$[-1,1]$上的Чебышев展开式为

3.15.1 $$f(x)=\sum_{k=0}^{\infty}c_kT_k(x)\qquad(|x|\leqslant 1)$$

其中$T_k(x)=\cos(k\ \arccos x)$ 为Чебышев多项式, 级数(3.15.1)是一致收敛和绝对收敛的, 在$[-1,1]$上用$R_{mn}(x)$做有理逼近, 有

3.15.2 $$R_{mn}(x)=\frac{p_m(x)}{Q_n(x)}=\frac{a_0T_0(x)+a_1T_1(x)+\cdots+a_mT_m(x)}{b_0T_0(x)+b_1T_1(x)+\cdots+b_nT_n(x)}$$

这里系数a_0, …, a_m及b_0, …, b_n是根据padé逼近的思想由下列$m+n+1$个方程确定的非零解:

3.15.3 $$\begin{cases} a_0 = b_0 c_0 + \dfrac{1}{2}\sum_{r=1}^{n} b_r c_r \\ a_k = b_0 c_k + \dfrac{1}{2} b_k c_0 + \dfrac{1}{2}\sum_{r=1}^{n} b_r (c_{r+k} + c_{|r-k|}) \end{cases}$$

$$(k=1,\cdots,m+n)$$

其中为书写方便，假定了当 $k>m$ 时取 $a_k=0$，当$k>n$时，取 $b_k=0$。由于方程组（3.15.3）中未知数个数多于方程个数，故非零解一定存在。利用 Чебышев 多项式的恒等性质

$$T_r(x)T_k(x) = \frac{1}{2}\left(T_{r+k}(x) + T_{|r-k|}(x) \right)$$

可以证明满足方程（3.15.3）的任一组解a_0，…，a_m及 b_0，…，b_n所得的 Чебышев 级数$P_m(x) - Q_n(x)f(x)$的前$m+n+1$项系数为 0。记

$$P_m(x) - Q_n(x)f(x) = \sum_{k=m+n+1}^{\infty} h_k T_k(x)$$

则

3.15.4 $$h_{m+n+1} = c_{m+n+1} + \frac{1}{2}\sum_{i=1}^{n} b_i (c_{m+n+i+1} + c_{m+n-i+1})$$

误差函数为

$$E_n(x) = R_{mn}(x) - f(x) = \frac{1}{Q_n(x)} \sum_{k=m+n+1}^{\infty} h_k T_k(x)$$

通常，由于取 $b_0=1$，量 h_k下降很快，所以

$$E_n(x) \approx h_{m+n+1} T_{m+n+1}(x)$$

因而$R_{mn}(x)$几乎是 f在$[-1,1]$上的最佳有理逼近。

实际计算时，常取$m=n$，$b_0=1$，这时（3.15.3）变成

$$
3.15.5 \quad \begin{cases} a_0 = c_0 + \dfrac{1}{2}\sum\limits_{r=1}^{n} b_r c_r \\ a_k = c_k + \dfrac{1}{2} b_k c_0 + \dfrac{1}{2}\sum\limits_{r=1}^{n} b_r (c_{r+k} + c_{|r-k|}) \end{cases}
$$

$$
(k=1,\cdots,n)
$$

$$
3.15.6 \quad \frac{1}{2}\sum_{r=1}^{n}(c_{n+k+r}+c_{n+k-r})b_r + c_{n+k}=0 \quad (k=1,\cdots,n)
$$

由（3.15.6）求出解b_1，…，b_n，代入（3.15.5）可求出a_0，…，a_n。为了估计误差，由（3.15.4）有

$$
h_{2n+1}=c_{2n+1}+\frac{1}{2}\sum_{i=1}^{n}b_i(c_{2n+i+1}+c_{2n-i+1})
$$

将此式与（3.15.6）联立，应用Gram（格拉姆）法则可求得

$$
b_0=\frac{h_{2n+1}\Delta_n}{D_n}=1, \quad 即 \quad h_{2n+1}=\frac{D_n}{\Delta_n}
$$

其中

$$
D_n=\begin{vmatrix} \frac{1}{2}(c_1+c_{2n+1}) & \frac{1}{2}(c_2+c_{2n}) & \cdots & \frac{1}{2}(c_n+c_{n+2}) & c_{n+1} \\ \frac{1}{2}(c_2+c_{2n+2}) & \frac{1}{2}(c_3+c_{2n+1}) & \cdots & \frac{1}{2}(c_{n+1}+c_{n+3}) & c_{n+2} \\ \cdots & \cdots & \cdots & & \cdots \\ \frac{1}{2}(c_{n+1}+c_{3n+1}) & \frac{1}{2}c_{n+2}+c_{3n}) & \cdots & \frac{1}{2}(c_{2n}+c_{2n+2}) & c_{2n+1} \end{vmatrix}
$$

Δ_n是行列式D_n去掉最后一行及最后一列的代数补式。

如果函数f具有奇偶性，为提高逼近精度可做变换$z=x^2$，若f为偶函数，这时

$$
f(x)=\sum_{k=0}^{\infty}c_k T_{2k}(x) \quad (|x|\leqslant 1)
$$

则

$$R_{mn}(x)=\frac{\sum_{k=0}^{m}a_kT_{2k}(x)}{\sum_{k=0}^{n}b_kT_{2k}(x)}$$

其中系数a_k及 b_k可用（3.15.3）的方法确定，当 f 为奇函数时

$$f(x)=\sum_{k=0}^{\infty}c_kT_{2k+1}(x)$$

可改写成

$$\frac{f(x)}{x}=\frac{1}{2}c_0^*+\sum_{k=1}^{\infty}c_k^*T_{2k}(x)$$

其中

$$c_k^*=2\sum_{j=0}^{\infty}(-1)^jc_{k+j}\quad,\qquad 2c_k=c_k^*+c_{k+1}^*$$

然后用 Maehly 方法求$x^{-1}f(x)$的有理逼近$R_{mn}(x)$。

3.15.7 **例** 求$f(x)=\arctan x$ 在 $|x|\leqslant 1$ 的 Maehly 逼近。

解 由于 $\arctan x$ 为奇函数，且

$$\arctan x=2\sum_{k=0}^{\infty}(-1)^k\frac{(\sqrt{2}-1)^{2k+1}}{2k+1}T_{2k+1}(x)\qquad(|x|\leqslant 1)$$

$$x^{-1}\arctan x=\frac{1}{2}c_0^*+\sum_{k=1}^{\infty}c_k^*T_{2k}(x)$$

其中

$$\frac{1}{2}c_0^*=\sum_{j=0}^{\infty}(-1)^jc_j=2\sum_{j=0}^{\infty}\frac{(\sqrt{2}-1)^{2j+1}}{2j+1}$$

$$=\ln(\sqrt{2}+1)=0.881373587019543$$

再由递推公式

$$c_{k+1}^* = 2c_k - c_k^* \quad (k=0,1,\cdots)$$

可以算出

$c_1^* = -0.105892924546706$， $c_2^* = 0.011135842059403$

$c_3^* = -0.001381195003598$， $c_4^* = 0.000185742973276$

$c_5^* = -0.000026215196110$， $c_6^* = 0.000003821036590$

$c_7^* = -0.000000569918604$

当 $m=n=2$ 时，由方程（3.15.6）得

$$\begin{cases} (c_2^* + c_4^*)b_1 + (c_1^* + c_5^*)b_2 = -2c_3^* \\ (c_3^* + c_5^*)b_1 + (c_2^* + c_6^*)b_2 = -2c_4^* \end{cases}$$

解此方程得

$b_1 = 0.37360540753$， $b_2 = 0.01385410972$

代入（3.15.5）得

$a_0 = 0.8616696410$， $a_1 = 0.2247301252$

$a_2 = 0.0033086795$

于是

$$\arctan x \approx \frac{x[a_0 + a_1 T_2(x) + a_2 T_4(x)]}{1 + b_1 T_2(x) + b_2 T_4(x)} = \frac{x[\alpha_0 + \alpha_1 x^2 + \alpha_2 x^4]}{\beta_0 + \beta_1 x^2 + \beta_2 x^4}$$

其中

$\alpha_0 = 0.6402481953$，$\alpha_1 = 0.4229908144$，$\alpha_2 = 0.0264694361$

$\beta_0 = 0.6402487022$，$\beta_1 = 0.6363779373$，$\beta_2 = 0.0108328778$

由（3.15.4）可得

$$h_5 = c_5^* + \frac{1}{2}(c_4^* + c_6^*)b_1 + \frac{1}{2}(c_3^* + c_7^*)b_2 \approx 3 \times 10^{-7}$$

于是误差估计为：

$$E_5(x) \approx h_5 x T_{10}(x)$$

3.16 函数的连分式展开

函数的连分式展开也是获得函数有理逼近的一种简单方

法。同幂级数展开一样，如果$f(\cdot)$在原点解析，则函数可展成连分式，展开方法可以利用$f(\cdot)$的幂级数展开式推出，也可利用$f(\cdot)$满足的常微分方程求其连分式解得到。

利用幂级数展开常用方法如下：

$$\frac{p_0+p_1x+p_2x^2+\cdots}{q_0+q_1x+q_2x^2+\cdots}=\frac{p_0}{q_0}+\frac{p_0+p_1x+p_2x^2+\cdots}{q_0+q_1x+q_2x^2+\cdots}-\frac{p_0}{q_0}$$

$$=\frac{p_0}{q_0}+\frac{x(p_0'-p_1'x+p_2'x^2+\cdots)}{q_0+q_1x+q_2x^2+\cdots}$$

$$=\frac{p_0}{q_0}+\cfrac{x}{\cfrac{q_0+q_1x+q_2x^2+\cdots}{p_0'+p_1'x+p_2'x^2+\cdots}}$$

3.16.1 例 $e^x=1+x+\frac{1}{2!}x^2+\frac{1}{3!}x^3+\cdots$

$$=\cfrac{1}{1+\cfrac{1}{1+x+\frac{1}{2!}x^2+\cdots}-1}$$

$$=\cfrac{1}{1-\cfrac{x(1+x/2!+x^2/3!+\cdots)}{1+x+\frac{x^2}{2!}+\frac{x^3}{3!}+\cdots}}$$

$$=\cfrac{1}{1-\cfrac{x}{1+\cfrac{1+x+x^2/2!+x^3/3!+\cdots}{1+\frac{x}{2!}+\frac{x^2}{3!}+\cdots}-1}}$$

$$=\cfrac{1}{1-\cfrac{x}{1+\cfrac{x(1/2+x/3+x^2/8+\cdots)}{1+\frac{1}{2}x+\frac{1}{6}x^2+\cdots}}}$$

$$=\cfrac{1}{1-\cfrac{x}{1+\cfrac{x}{2+\cfrac{1+\frac{x}{2}+\frac{x^2}{6}+\cdots}{\frac{1}{2}+\frac{x}{3}+\frac{x^2}{8}+\cdots}-2}}}=\cdots$$

3.16.2 例 $\tan x=\dfrac{x-\dfrac{x^3}{3!}+\dfrac{x^5}{5!}-\cdots}{1-\dfrac{x^2}{2!}+\dfrac{x^4}{4!}-\cdots}$

$$=\cfrac{x}{1+\cfrac{1-\frac{x^2}{2!}+\frac{x^4}{4!}-\cdots}{1-\frac{x^2}{3!}+\frac{x^4}{5!}-\cdots}-1}$$

$$=\cfrac{x}{1+\cfrac{-x^2\left(\frac{1}{3}-\frac{x^2}{30}+\cdots\right)}{1-\frac{x^2}{3!}+\frac{x^4}{5!}-\cdots}}$$

$$=\cfrac{x}{1-\cfrac{x^2}{3+\cfrac{1-x^2/3!+x^4/5!-\cdots}{\frac{1}{3}-\frac{x^2}{30}+\frac{x^4}{7\cdot 5!}-\cdots}-3}}$$

$$=\cfrac{x}{1-\cfrac{x^2}{3-\cfrac{x^2}{5+\cfrac{1-x^2/10+\cdots}{\frac{1}{5}-\frac{x^2}{70}+\cdots}-5}}}$$

$=\cdots\cdots$

利用幂级数求函数的连分式展开通常只能得到前若干项系数，要得到 $f(x)$ 连分式展开的一般项可用常微分方程连分式解法。设微分方程

3.16.3 $\quad y' = f(x, y) \quad, y(0) = 0$

当 x 很小时可求得 $y_0 \approx \xi_0(x)$，令 $y = \dfrac{\xi_0(x)}{1 + y_1}$ 为方程(3.16.3)的解。y_1 适合微分方程 $y_1' = f_1(x_1, y_1)$，$y_1(0) = 0$，它的解 $y_1 = \dfrac{\xi_1(x)}{1 + y_2}$，重复这一步骤可得（3.16.3）的连分式解为

$$y = \frac{\xi_0(x)}{1} + \frac{\xi_1(x)}{1} + \cdots + \frac{\xi_n(x)}{1} + \cdots$$

为使 $\xi_v(x)$ 计算有规律，通常要使 f 与 f_1 有相同形式，这样 $\xi_0(x)$ 与 $\xi_1(x)$ 也有相同形式，并可通过逐次递推得到，通常取 $\xi_v(x) = a_v xk_v$ 的形式，其中 $k_v \geq 0$。

方程

3.16.4 $$\begin{cases} (1 + \eta x^k)\dfrac{xy'}{k} + (\beta + \alpha x^k)y + \gamma y^2 = \delta x^k \\ y(0) = 0 \end{cases}$$

其中 k，η，β，α，γ，δ 均为常数，且 $k > 0$。这是一种特殊的 Riccati（黎卡提）方程，很多基本初等函数的连分式展开可利用这个方程的解得到。用连分式求方程（3.16.4）的解，在 $|x| \ll 1$ 时，方程高阶项可忽略不计，(3.16.4)可简化为线性方程

$$\frac{xy_0'}{k} + \beta y_0 = \delta x^k \quad , \quad y_0(0) = 0$$

此方程的解 $\quad y_0(x) = \dfrac{\delta x^k}{1 + \beta} = \xi_0(x)$。

令方程（3.16.4）的解为 $\quad y(x) = \dfrac{\delta x^k}{1 + \beta + y_1}$

则

$$y' = \frac{k\delta x^{k-1}(1+\beta+y_1) - \delta x^k y_1'}{(1+\beta+y_1)^2}$$

将它代入（3.16.4）化简，并令 $\beta_1 = 1+\beta$，$\alpha_1 = -(\eta+\alpha)$，$\gamma_1 = 1$，$\delta_1 = \gamma\delta + (1+\beta)(\eta+\alpha)$，可得到

3.16.5 $$(1+\eta x^k)\frac{xy_1'}{k} + (\beta_1 + \alpha_1 x^k)y_1 + \gamma_1 y_1^2 = \delta_1 x^k$$

$$y_1(0) = 0$$

与解方程（3.16.4）类似，可得方程（3.16.5）的解为

$$y_1 = \frac{\delta_1 x^k}{1+\beta_1+y_2}$$

代入（3.16.5）得到关于y_2的微分方程

$$(1+\eta x^k)\frac{xy_2'}{k} + (\beta_2 + \alpha_2 x^k)y_2 + \gamma_2 y_2^2 = \delta_2 x^k, \; y_2(0) = 0$$

其中 $\beta_2 = 1+\beta_1 = 2+\beta$，$\alpha_2 = -(\eta+\alpha_1) = \alpha$，$\gamma_2 = 1$

$\delta_2 = \gamma_1\delta_1 + (1+\beta_1)(\eta+\alpha_1) = \gamma\delta + (1+\beta)\eta - \alpha$

依此类推，有

$$y_{n-1} = \frac{\delta_{n-1} x^k}{1+\beta_{n-1}+y_n}$$

$$(1+\eta x^k)\frac{xy_n'}{k} + (\beta_n + \alpha_n x^k)y_n + \gamma_n y_n^2 = \delta_n x^k$$

$y_n(0) = 0$

其中

3.16.6 $$\begin{cases} \beta_n = 1+\beta_{n-1}, \quad \alpha_n = -(\eta+\alpha_{n-1}), \quad \gamma_n = 1 \\ \delta_n = \gamma_{n-1}\delta_{n-1} + (1+\beta_{n-1})(\eta+\alpha_{n-1}), \quad n = 2, 3, \cdots \end{cases}$$

于是方程（3.16.4）的连分式解为

3.16.7 $$y = \frac{\delta x^k}{1+\beta} + \frac{\delta_1 x^k}{1+\beta_1} + \cdots + \frac{\delta_n x^k}{1+\beta_n} + \cdots$$

其中 β_1，β_2，$\cdots$，δ_1，δ_2，$\cdots$可由递推公式（3.16.6）得到

$\beta_m = m+\beta \qquad (m = 1, 2, \cdots)$

$$\delta_m=\gamma_1\delta_1+\sum_{j=1}^{m-1}(1+\beta_j)(\eta+\alpha_j)\quad(m\geqslant 2)$$

在 $m=2n$ 及 $m=2n+1$ 的情况下可得到

$$\delta_{2n}=\gamma\delta+(n+\beta)n\eta-n\alpha\qquad(n=1,2,\cdots)$$

$$\delta_{2n+1}=\gamma\delta+(\beta+n+1)[(n+1)\eta+\alpha]\qquad(n=0,1,\cdots)$$

将以上结果代入（3.16.7）就得到(3.16.4)的解

3.16.8
$$y=\frac{\delta x^k}{1+\beta}+\frac{[\gamma\delta+(1+\beta)(\eta+\alpha)]x^k}{2+\beta}+\frac{[\gamma\delta+(1+\beta)\eta-\alpha]x^k}{3+\beta}+\cdots+\frac{[\gamma\delta+(\beta+n)(n\eta+\alpha)]x^k}{2n+\beta}+\frac{[\gamma\delta+(n+\beta)n\eta-n\alpha]x^k}{2n+1+\beta}+\cdots$$

对于函数 $y=\tan x$，满足微分方程

3.16.9 $\quad y'=1+y^2,\quad y(0)=0$

令 $\quad y=\dfrac{x}{1+z},\quad z(0)=0$

将它代入（3.16.9）得到关于 z 的微分方程

3.16.10
$$\frac{xz'}{2}+\frac{1}{2}z+\frac{1}{2}z^2=-\frac{1}{2}x^2,\qquad z(0)=0$$

与方程（3.16.4）比较得

$$\eta=0,\quad k=2,\quad \beta=\gamma=\frac{1}{2},\quad \alpha=0,\ \delta=-\frac{1}{2}$$

故利用（3.16.8）可得（3.16.10）的连分式解为

3.16.11
$$z=\frac{-\frac{1}{2}x^2}{3/2}+\frac{-\frac{1}{4}x^2}{5/2}+\frac{-\frac{1}{4}x^2}{7/2}+\cdots$$

$$\begin{array}{l} -\frac{1}{4}x_2 \\ \hline \end{array}$$

$$+\frac{-\frac{1}{4}x_2}{\frac{2n+1}{2}}+\cdots$$

$$=\frac{-x_2}{3}-\frac{x^2}{5}-\frac{x^2}{7}-\cdots-\frac{x^2}{2n+1}+\cdots$$

于是，$y=\tan x$的连分式展开为

$$3.16.12\quad \tan x=\frac{x}{1}-\frac{x^2}{3}-\frac{x^2}{5}-\frac{x^2}{7}-\cdots-\frac{x^2}{2n+1}-\cdots$$

由 $\operatorname{th}x=i\tan\frac{x}{i}$，用$\frac{x}{i}$代替上式的$x$并乘$i$得

$$\operatorname{th}x=\frac{x}{1}+\frac{x^2}{3}+\frac{x^2}{5}+\cdots+\frac{x^2}{2n+1}+\cdots$$

若考虑 $y=\arctan x$，满足微分方程

$$y'=\frac{1}{1+x^2},\qquad y(0)=0$$

令 $y=\frac{x}{1+z}$，$z(0)=0$，可得

$$3.16.13\quad (1+x^2)\frac{xz'}{2}+\left(\frac{1}{2}-\frac{1}{2}x^2\right)z+\frac{1}{2}z^2=\frac{1}{2}x^2$$

$z(0)=0$

与（3.16.4）比较得

$$\eta=1,\quad k=2,\quad \beta=\gamma=\delta=\frac{1}{2},\quad \alpha=-\frac{1}{2}$$

由（3.16.8）可得连分式解

$$3.16.14\quad z=\frac{x^2/2}{3/2}+\frac{\left(\frac{1}{4}+\frac{3}{4}\right)x^2}{5/2}+\frac{\left(\frac{1}{4}+2\right)x^2}{7/2}+\cdots$$

$$+\frac{\left[\frac{1}{4}+\frac{(2n+1)(2n-1)}{4}\right]x^2}{\frac{4n+1}{2}}$$

$$+\frac{\left[\frac{1}{4}+n^2+n\right]x^2}{\frac{4n+3}{2}}+\cdots$$

$$=\frac{x^2}{3}+\frac{4x^2}{5}+\frac{9x^2}{7}+\cdots+\frac{4n^2x^2}{4n+1}+\frac{(2n+1)^2x^2}{4n+3}+\cdots$$

于是得

3.16.15 $\arctan x=\frac{x}{1}+\frac{x^2}{3}+\frac{4x^2}{5}+\frac{9x^2}{7}+\cdots+\frac{n^2x^2}{2n+1}+\cdots$

由于 $\frac{1}{2}\ln\frac{1+x}{1-x}=\operatorname{arcth}x=i\arctan\frac{x}{i}$，所以有

3.16.16 $\frac{1}{2}\ln\frac{1+x}{1-x}=\frac{x}{1}-\frac{x^2}{3}-\frac{4x^2}{5}-\cdots-\frac{n^2x^2}{2n+1}-\cdots$

应用方程（3.16.4）的连分式解（3.16.8）还可得到下列基本初等函数的连分式展开：

3.16.17 $(1+x)^\nu=\frac{1}{1}-\frac{\nu x}{1}+\frac{(1+\nu)x}{2}+\frac{(1-\nu)x}{3}$

$$+\frac{(2+\nu)x}{2}+\frac{(2-\nu x)}{5}+\cdots$$

$$+\frac{(n+\nu)x}{2}+\frac{(n-\nu)x}{2n+1}+\cdots$$

3.16.18 $\ln(1+x)=\frac{x}{1}+\frac{x}{2}+\frac{x}{3}+\frac{2x}{2}+\frac{2x}{5}+\cdots+\frac{nx}{2}$

$$+\frac{nx}{2n+1}+\cdots$$

3.16.19 $e^{x}=\frac{1}{1}-\frac{x}{1}+\frac{x}{2}-\frac{x}{3}+\frac{x}{2}-\frac{x}{5}+\cdots+\frac{x}{2}$

$$-\frac{x}{2n+1}-\cdots$$

4

第 4 章　数值积分与数值微分

4.1

4.1 引　言

计算定积分

4.1.1 $$I(f)=\int_a^b f(x)dx$$

的值，这在计算中经常碰到.这个问题似乎很简单，只要求出f的原函数F就可以求得积分（4.1.1）的值，即$I(f)=F(b)-F(a)$，但是实际情况并不全是如此．其原因如下：

1° 有很多简单的积分并不能解析地求解。例如，积分$\int_0^1 \frac{1}{\ln x}dx$，$\int_0^\pi \frac{\sin x}{x}dx$，$\int_0^\pi \frac{1}{1-k^2\sin^2 x}dx$等都不能解析地求解.

2° 有些积分虽然可以解析地求解，但原函数比被积函数复杂得多，而且也难于给出最后的数值结果。例如，积分

$$I(f)=\int_{-1}^1 \frac{2x^2+2x+13}{(x-2)(x^2+1)^2}dx$$

经一系列运算，可以得到

$$I(f)=\left[\frac{1}{2}\ln\frac{(2-x)^2}{1+x^2}-4\arctan x+\frac{3-4x}{2(1+x^2)}\right]\Bigg|_{-1}^{1}$$

$$=\ln\frac{1}{3}-8\arctan 1-2$$

3° 有不少情况，被积函数f没有具体的表达式，因此无法采用解析求解。

因此，采用数值方法来计算定积分的值是非常必要的。对于由函数表格形式给出的函数，要求其导数时，也只能用数值方法近似地求解。

4.2

4.2 Newton-Cotes求积公式

4.2.1 公式的一般形式

将积分(4.1.1)中的积分区间$[a, b]$分成n等分，其节点x_k为

$$x_k=a+kh\ ,\ h=\frac{1}{n}(b-a)\qquad(k=0,1,\cdots,n)$$

对于给定f，在节点$x_k(k=0, 1, \cdots, n)$上的值$f(x_k)$为已知.那么f在$n+1$个节点$x_0, x_1, \cdots, x_n$上的n次代数插值多项式为

$$P_n(x)=\sum_{k=0}^{n}\left(\prod_{\substack{j=0\\j\neq k}}^{n}\frac{x-x_j}{x_k-x_j}\right)f(x_k)$$

如果记$x=a+th$，则上式可以写为

$$P_n(x)=\sum_{k=0}^{n}\left(\prod_{\substack{j=0\\j\neq k}}^{n}\frac{t-j}{k-j}\right)f(x_k)\tag{4.2.1}$$

在积分(4.1.1)中的被积函数f用其$n+1$个节点的代数插值多项式P_n来代替，可得

$$I(f)=\int_a^b f(x)dx\approx I_n(f)=\int_a^b P_n(x)dx$$

多项式的积分是容易求出的，因此把上式写为

$$I(f)\approx I_n(f)=\sum_{k=0}^{n}A_kf(x_k)\tag{4.2.2}$$

其中

4.2.3 $A_k=\frac{b-a}{n}\int_0^n \prod_{\substack{j=0\\ j\neq k}}^{n}\frac{t-j}{k-j}dt=(b-a)c_k^{(n)}$

4.2.4 $c_k^{(n)}=\frac{(-1)^{n-k}}{k!(n-k)!n}\int_0^n \prod_{\substack{j=0\\ j\neq k}}^{n}(t-j)dt$

公式（4.2.2）称为 Newton-Cotes（牛顿-柯特斯）求积公式或称为等距节点求积公式，A_k称为求积公式系数，$c_k^{(n)}$称为Cotes 求积系数。

Newton-Cotes求积公式的误差估计 $E_n(f)=I(f)-I_n(f)$，由下面定理给出：

4.2.5 **定理** 1° 如果 n 为偶数，$f^{(n+2)}$ 在$[a,b]$上连续，则有

4.2.6 $E_n(f)=c_nh^{n+3}f^{(n+2)}(\eta)$ ，$\eta\in[a,b]$

其中

$$c_n=\frac{1}{(n+2)!}\int_0^n t^2(t-1)(t-2)\cdots(t-n)dt$$

2° 如果 n 为奇数，$f^{(n+1)}$ 在$[a,b]$上连续，则有

4.2.7 $E_n(f)=c_nh^{n+2}f^{(n+1)}(\eta)$，$\eta\in[a,b]$

其中

$$c_n=\frac{1}{(n+1)!}\int_0^n t(t-1)(t-2)\cdots(t-n)dt$$

从定理（4.2.5）可以看出，如果 f 是 n 次多项式，得 $f^{(n+1)}=0$，所以$E_n(f)=0$。由此可知，对于次数不高于 n的多项式来说，Newton-Cotes 公式（4.2.2）是精确成立的。

近似式

4.2.8 $\int_a^b f(x)dx\approx\sum_{k=0}^{n}A_kf(x_k)$

称为一般的求积公式，其中 x_k称为求积节点，A_k称为求积系

数，亦称伴随节点 x_k 的权(Weight)。A_k 仅仅与节点 x_k 的选取有关而不依赖于被积函数 $f(x)$ 的形式。

数值积分(Numerical Integration)是一个近似方法，因此对尽可能多的被积函数 f，要求数值积分能准确地作出计算，这就引出了代数精度的概念。

4.2.9 定义 如果求积公式（4.2.8）对所有次数不高于 n 次的代数多项式等式精确成立，但对于 $n+1$ 次的代数多项式等式不成立，则称求积公式（4.2.8）具有 n 次代数精度。

由定理（4.2.5)可知，Newton-Cotes 求积公式（4.2.2）的代数精度至少是 n 次，而当 n 是偶数时，（4.2.2）的代数精度可达 $n+1$ 次。

4.2.2 梯形公式

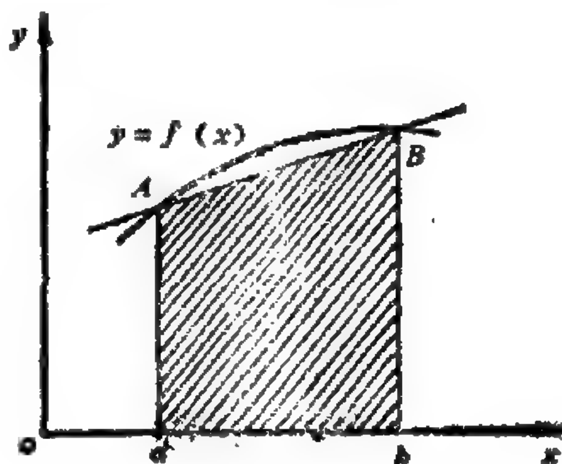
在 Newton-Cotes 公式（4.2.2）中，取 $n=1$ 时

$$c_0^{(1)}=c_1^{(1)}=\frac{1}{2}$$

所以有

4.2.10 $$I(f)\approx I_1(f)=\frac{b-a}{2}[f(a)+f(b)]$$

公式(4.2.10)称为**梯形公式**(Trapezoidal Rule)。如果用连接 $(a,f(a))$ 和 $f(b,(b))$ 的直线来逼近 f，并求积这直线公式可得到 $I_1(f)$。(图4.2.11)中 $I_1(f)$ 就是梯形 $AabB$ 的面积，因此用 $I_1(f)$ 来逼近 $I(f)$，就是用梯形面积来代替曲边梯形面积。



4.2.11 图

4.2.12 **定理** 若f在$[a, b]$上有二阶连续导数，则梯形求积公式（4.2.10）的误差估计为

$$E_1(f)=I(f)-I_1(f)=-\frac{1}{12}(b-a)^3 f''(\eta) \quad ,\eta\in[a,b]$$

4.2.3 Simpson 公式

在 Newton-Cotes 公式（4.2.2）中，取 $n=2$，则有

$$c_0^{(2)}=\frac{1}{4}\int_0^2 (t-1)(t-2)dt=\frac{1}{6}$$

$$c_1^{(2)}=-\frac{1}{2}\int_0^2 t(t-2)dt=\frac{4}{6}$$

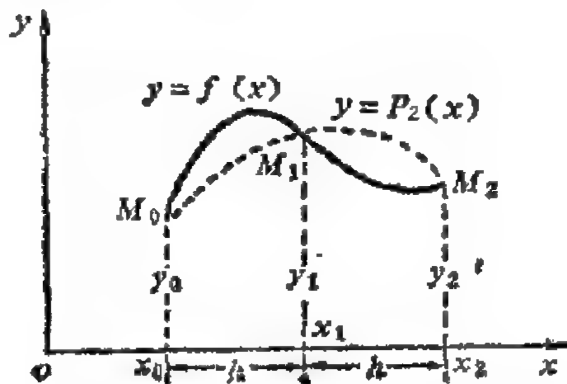
$$c_2^{(2)}=\frac{1}{4}\int_0^2 t(t-1)dt=\frac{1}{6}$$

由此得到

4.2.13 $$I(f)\approx I_2(f)=\frac{h}{3}\left[f(a)+4f\left(\frac{a+b}{2}\right)+f(b)\right]$$

其中 $h=\frac{1}{2}(b-a)$．（4.2.13）称为 Simpson（辛甫生）公式，其几何意义就是用通过三点 $M_0(x_0, y_0)$，$M_1(x_1, y_1)$，$M_2(x_2, y_2)$的抛物线 P_2围成的曲边形面积来代替给定函数f围成的曲边形面积（见图4.2.14）。由于上述原因，公式（4.2.13）也称抛物线公式．

4.2.14 **图**



4.2.15 定理 若f在$[a, b]$上有四阶连续导数，则Simpson公式（4.2.13）的误差估计为

$$E_2(f)=I(f)-I_2(f)=-\frac{1}{90}h^5 f^{(4)}(\eta)\quad, \eta\in[a,b]$$

4.2.4 高阶 Newton-Cotes 公式

在 Newton-Cotes 公式（4.2.2)中，$n\geqslant 3$ 的公式称为**高阶 Newton-Cotes 公式**.

对于 $n=3$，$h=\frac{1}{3}(b-a)$，可以求得

4.2.16 $I(f)\approx I_3(f)$

$$=\frac{3}{8}h[f(a)+3f(a+h)+3f(b-h)+f(b)]$$

此公式称为**Simpson 3/8 公式**. 如果$f^{(4)}$在$[a, b]$上连续，则公式（4.2.16）的误差估计有

4.2.17 $E_3(f)=I(f)-I_3(f)=-\frac{3}{80}h^5 f^{(4)}(\eta)\quad, \eta\in[a,b]$

在 Newton-Cotes 公式中，取$n=4$，$h=\frac{1}{4}(b-a)$，则公式化为

4.2.18 $I(f)\approx I_4(f)=\frac{2}{45}h\Big[7f(a)+32f(a+h)$

$$+12f\left(\frac{a+b}{2}\right)+32f(b-h)+7f(b)\Big]$$

通常把这个公式称作**Cotes 公式**. 如果$f^{(6)}$在$[a,b]$上连续，那么公式（4.2.18）的误差估计是

4.2.19 $E_4(f)=-\frac{8}{945}h^7 f^{(6)}(\eta)$，$\eta\in[a,b]$

根据 Cotes 系数公式（4.2.4)，有 Cotes 系数表（$n\leqslant 8$）.

4.2.20 表

n	$c_k^{(n)}$								
1	$\frac{1}{2}$	$\frac{1}{2}$							
2	$\frac{1}{6}$	$\frac{4}{6}$	$\frac{1}{6}$						
3	$\frac{1}{8}$	$\frac{3}{8}$	$\frac{3}{8}$	$\frac{1}{8}$					
4	$\frac{7}{90}$	$\frac{16}{45}$	$\frac{2}{15}$	$\frac{16}{45}$	$\frac{7}{90}$				
5	$\frac{19}{288}$	$\frac{25}{96}$	$\frac{25}{144}$	$\frac{25}{144}$	$\frac{25}{96}$	$\frac{19}{288}$			
6	$\frac{41}{840}$	$\frac{9}{35}$	$\frac{9}{280}$	$\frac{34}{105}$	$\frac{9}{280}$	$\frac{9}{35}$	$\frac{41}{840}$		
7	$\frac{751}{17280}$	$\frac{3577}{17280}$	$\frac{1323}{17280}$	$\frac{2989}{17280}$	$\frac{2989}{17280}$	$\frac{1323}{17280}$	$\frac{3577}{17280}$	$\frac{751}{17280}$	
8	$\frac{989}{28350}$	$\frac{5888}{28350}$	$\frac{-928}{28350}$	$\frac{10496}{28350}$	$\frac{-4540}{28350}$	$\frac{10496}{28350}$	$\frac{-928}{28350}$	$\frac{5888}{28350}$	$\frac{989}{28350}$

从误差分析来考虑，好象 n 越大，精度越高。但由于高阶 Newton-Cotes 求积公式的舍入误差的干扰比较大，因此，在实际计算中一般不宜采用 n 很大的 Newton-Cotes 公式。

4.2.5 开型 Newton-Cotes 公式

令 $h=(b-a)/(n+2)$，$x_0=a+h$，$x_k=x_0+kh$ $(k=1,2,\cdots,n)$ 由此知，$x_n=b-h$。用 $x_{-1}=a$，$x_{n+1}=b$ 表示积分区间$[a,b]$的端点，求积公式可写为

4.2.21 $I(f)=\int_{x_{-1}}^{x_{n+1}} f(x)dx \approx \sum_{k=0}^{n} A_k f(x_k)$

其中 A_k，$k=0, 1, \cdots, n$，由公式（4.2.3）给出。

注意：在上述求积公式中，积分区间的端点皆没有采用。

如果在求积公式（4.2.2）中，积分区间的端点至少有一个不用，则称（4.2.2）为**开型 Newton-Cotes 求积公式**。显然，公式(4.2.21)是开型 Newton-Cotes 公式，而(4.2.21)以前的 Newton-Cotes 公式称为**闭型公式**。

开型 Newton-Cotes 求积公式的误差估计有下面定理：

4.2.22 定理 设 $\sum_{k=0}^{n} A_k f(x_k)$ 表示 $n+1$个节点的开型 Newton-Cotes 求积公式，其中 $x_{-1}=a$，$x_{n+1}=b$，$h=(b-a)/(n+2)$。

1° 如果 n 为偶数，$f^{(n+2)}$ 在$[a, b]$上连续，则有

4.2.23 $E_n(f)=c_n h^{n+3} f^{(n+2)}(\eta)$，$\eta\in[a,b]$

其中

$$c_n=\frac{1}{(n+2)!}\int_{-1}^{n+1} t^2(t-1)\cdots(t-n)dt$$

2° 如果 n 为奇数，$f^{(n+1)}$ 在$[a, b]$上连续，则有

4.2.24 $E_n(f)=c_n h^{n+2} f^{(n+1)}(\eta)$，$\eta\in[a,b]$

其中

$$c_n=\frac{1}{(n+1)!}\int_{-1}^{n+1} t(t-1)\cdots(t-n)dt$$

在实际计算中，闭型公式一般要比同阶的开型公式更为精确，因此经常使用的是闭型 Newton-Cotes 求和公式。但在 $n=0$ 时，开型 Newton-Cotes 公式是经常使用的，其求积公式为

4.2.25 $I(f)\approx I_0(f)=2hf(x_0)$

其中　$h=\frac{1}{2}(b-a)$，$x_0=\frac{1}{2}(a+b)$

公式（4.2.25）称为**中点公式**（Midpoint Rule）．如果f在$[a, b]$上有二阶连续导数，则中点公式的误差估计为

4.2.26　$E_0(f)=\frac{1}{3}h^3f''(\eta),\ \eta\in[a,b]$

4.3

4.3　复化求积公式

在比较大的积分区间上采用低阶 Newton-Cotes 公式进行计算，精度较低，而采用高阶 Newton-Cotes 公式计算，舍入误差影响较大．此外，Newton-Cotes 积分法是基于在等距点上的插值多项式，有些函数的高阶插值多项式有振荡，由此得到的求积不正确，所以一般不采用阶数很高的 Newton-Cotes 求积公式．为改善求积的精度，通常采用**复化求积法**（Composite Numerical Integration）．

将积分区间$[a, b]$分为n等分，步长$h=\frac{1}{n}(b-a)$，分点为$x_k=a+kh$，$k=0, 1, \cdots, n$．复化求积法的步骤是：先用低阶 Newton-Cotes 公式求得在每个子区间$[x_k, x_{k+1}]$上的积分近似值，然后对这些近似值求和，从而得到$I(f)$的近似值．

4.3.1　复化梯形公式

在每个小区间$[x_k, x_{k+1}]$上使用梯形求积公式，再求和得到逼近$I(f)$的求积公式

4.3.1　$I(f)\approx T_n=\frac{1}{2}h\sum_{k=0}^{n-1}[f(x_k)+f(x_{k+1})]$

$$=\frac{1}{2}h\left[f(a)+2\sum_{k=1}^{n-1}f(x_k)+f(b)\right]$$

公式（4.3.1）称为**复化梯形公式**(Composite Trapezoidal Rule)，T_n的下标n表示积分区间$[a, b]$分成n等分。

对每个小区间$[x_k, x_{k+1}]$上的梯形公式进行误差估计，然后相加就能得到复合梯形公式的误差估计。

4.3.2 **定理** 如果f在$[a, b]$上有二阶连续导数，那么复化梯形公式（4.3.1）的误差估计为

$$E_n(f)=I(f)-T_n=-\frac{(b-a)}{12}h^2f''(\eta),\eta\in[a,b]$$

4.3.2 复化 Simpson 公式

类似于复化梯形公式，但在小区间$[x_k, x_{k+1}]$上不用梯形公式而采用Simpson公式，可以得到逼近$I(f)$的求积公式

4.3.3 $$I(f)\approx S_n=\frac{h}{6}\sum_{k=0}^{n-1}[f(x_k)+4f(x_{k+\frac{1}{2}})+f(x_{k+1})]$$

其中 $x_{k+\frac{1}{2}}=\frac{1}{2}(x_k+x_{k+1})$。公式（4.3.3）称为**复化Simpson公式**，S_n的下标n表示把积分区间$[a, b]$分为n等分。

如果令

$$H_n=h\sum_{k=0}^{n-1}f(x_{k+\frac{1}{2}})=h\sum_{k=1}^{n}f\left(a+(2k-1)\frac{b-a}{2n}\right)$$

那么容易得到复化Simpson公式和复化梯形公式之间的关系

$$S_n=\frac{1}{3}T_n+\frac{2}{3}H_n$$

4.3.4 **定理** 如果f在$[a, b]$上有四阶连续导数，那么复化Sim-

pson 公式 (4.3.3) 的误差估计为

$$E_n(f)=I(f)-S_n=-\frac{b-a}{2880}h^4f^{(4)}(\eta),\ \eta\in[a,b]$$

4.3.5 例 分别用复化梯形公式和复化 Simpson 公式计算积分

$$I(f)=\int_0^{\pi}e^x\cos(x)dx$$

解 积分$I(f)$的精确值为$I(f)=-12.0703463164$.

T_n,S_n以及它们的误差 $E_n^{(t)}$, $E_n^{(s)}$见表 (4.3.6).

4.3.6 表

n	T_n	$E_n^{(t)}$	S_n	$E_n^{(s)}$
2	−17.389259	5.32	−11.5928395534	-4.78×10^{-1}
4	−13.336023	1.27	−11.9849440198	-8.54×10^{-2}
8	−12.382162	3.12×10^{-1}	−12.0642089572	-6.14×10^{-3}
16	−12.148004	7.77×10^{-2}	−12.0699513233	-3.95×10^{-4}
32	−12.089742	1.94×10^{-2}	−12.0703214561	-2.49×10^{-5}
64	−12.075194	4.85×10^{-3}	−12.0703447599	-1.56×10^{-6}
128	−12.071558	1.21×10^{-3}	−12.0703462191	-9.73×10^{-8}
256	−12.070649	3.03×10^{-4}	−12.0703463103	-6.08×10^{-9}

复化 Simpson 公式比复化梯形公式精确得多，并随 n 的增大，误差减少得快。

4.3.3 复化求积公式的收敛性

对于Newton－Cotes公式来说，当$n\to\infty$ 时并不能保证$I_n(f)\to I(f)$.如定积分

$$I(f)=\int_{-4}^{4}\frac{dx}{1+x^2}\approx2.6516$$

利用Newton－Cotes公式计算结果见表（4.3.7）。

4.3.7 表

n	$I_n(f)$
2	5.4902
4	2.2776
6	3.3288
8	1.9411
10	3.5956

可以看出$I_n(f)$是发散的。

对于复化求积公式来说，情况大不相同。可以把复化梯形公式T_n和复化Simpson公式S_n看作Riemann（黎曼）和，因此由积分$I(f)=\int_a^b f(x)dx$的存在性可以得到T_n和S_n的收敛性，其收敛到$I(f)$。收敛快慢的判别有定义（4.3.8）。

4.3.8 **定义** 设复化求积公式为$I(f)\approx I_n$，如果当$h\to 0$时有

$$\frac{I(f)-I_n}{h^P}\to C$$

其中C为一个非零常数，那么称I_n是P阶收敛的。

由定理（4.3.2）及定理（4.3.4）知，复化梯形公式是二阶收敛的，复化Simpson是四阶收敛的。

4.3.4 区间逐次分半法

利用复化梯形公式和复化Simpson公式来进行定积分的近似计算是比较简单的，但是为了确定把积分区间$[a,b]$分成多少个子区间，即n取多大，则需依据余项公式作事先估计，就要分析被积函数的高阶导数，而这是很困难的。

区间逐次分半法 就是根据规定的精度要求，在计算过程中把积分区间逐次分半，并利用前后两次计算结果来判别误

差的大小，从而得到满足精度要求的积分近似值.

利用复化梯形公式的误差估计（定理4.3.2）可以得到

4.3.9 $\frac{1}{3}(T_{2n}-T_n)\approx I(f)-T_{2n}$

上式提供了一个误差估计的判别条件．如果

$|T_{2n}-T_n|<\varepsilon$（允许误差）

那么可以认为T_{2n}已满足精度要求．

需要特别注意的是，在逐次分半过程中，老分点上的函数值要避免计算．

区间逐次分半法的具体计算过程如下：

最初取$n=1$，计算

$$T_1=h_0\left[\frac{f(a)}{2}+\frac{f(b)}{2}\right],\ h_0=b-a$$

然后将区间分半，取$n=2$，计算

$$T_2=h_1\left[\frac{f(a)}{2}+\frac{f(b)}{2}+f(x_1)\right]=\frac{1}{2}T_1+h_1f(x_1)$$

其中$h_1=\frac{1}{2}(b-a)$，$x_1=a+h_1$

再将每个小区间分半，则$n=4$，计算

$$T_4=h_2\left[\frac{f(a)}{2}+\frac{f(b)}{2}+f(x_1)+f(x_2)+f(x_3)\right]$$

$$=\frac{1}{2}T_2+h_2[f(x_1)+f(x_3)]$$

其中$h_2=\frac{1}{4}(b-a)$，$x_k=a+kh_2\quad(k=1,2,3)$

一般地，每次总是在前一次的基础上再将小区间分半，则分点加密．一般计算公式为

4.3.10 $T_{2n}=\frac{1}{2}T_n+h_{2n}\sum_{k=1}^{n}f[a+(2k-1)h_{2n}]$

其中 $h_{2n}=\frac{1}{2n}(b-a)$

利用公式（4.3.10）计算出T_{2n}后，再进行检验不等式

4.3.11 $|T_{2n}-T_n|<\varepsilon$（取绝对误差）

或

4.3.12 $\frac{|T_{2n}-T_n|}{|T_{2n}|}<\varepsilon$（取相对误差）

是否满足，如果满足则取T_{2n}为所求定积分之近似值，否则区间继续分半，重复上述过程直至条件满足。一般在实际计算中，当$n>n_0$（n_0为给定一正整数）时才进行条件(4.3.11)或条件(4.3.12) 的判别，否则可能出现假收敛。

对于Simpson公式，也可以同样进行区间逐次分半。由复化Simpson公式的误差估计(定理4.3.4) 可以得到

4.3.13 $\frac{1}{15}(S_{2n}-S_n)\approx[I(f)-S_{2n}]$

因此可以由$S_{2n}-S_n$来估计S_{2n}的误差。用区间逐次分半产生复化Simpson求积的序列

$S_1,S_2,S_4,\cdots,S_n,S_{2n},\cdots$

其中

4.3.14 $S_n=\frac{1}{3}h_n[f(a)+f(b)+2P_n+4Q_n]$

$(n=1,2,4,\cdots)$

P_n是在老分点上函数值之和，

$$P_n=\sum_{k=1}^{n-1}f(x_{2k})$$

Q_n是在新分点上函数值之和

$$Q_n=\sum_{k=1}^{n}f(x_{2k-1})$$

$$Q_n=\frac{1}{2n}(b-a),\quad x_k=a+kh_n\ (k=1,2,\cdots,2n-1)$$

在老分点上的函数值不需重复计算.

利用公式（4.3.14）计算出S_{2n}之后，再进行精度检验.如果满足条件

4.3.15 $|S_{2n}-S_n|<\varepsilon$（取绝对误差）

或

4.3.16 $\frac{|S_{2n}-S_n|}{|S_{2n}|}<\varepsilon$（取相对误差）

那么取S_{2n}作为$I(f)$的近似值，否则再分半区间继续进行.

4.4 Richardson外推算法和Romberg积分法

利用复化梯形求积公式来计算积分$I(f)$的近似值，精度较差.当$n\to\infty$，序列$T_1,T_2,\cdots,T_n\cdots$收敛到$I(f)$，但速度缓慢，为加速$\{T_n\}$的收敛速度，可利用外推（Extrapolation）算法来加速.

4.4.1 Richardson外推算法

设函数S在$x=0$处的值$S(0)$由序列$S(h),S\left(\frac{h}{2}\right),\cdots(h>0)$来逼近.通过序列$S(h)$，$S\left(\frac{h}{2}\right)$，…构造出一个新的序列，使它更快地收敛于$S(0)$.运用Taylor（泰勒）展开式

$$S(h)=S(0)+hS'(0)+\frac{1}{2!}h^2S''(0)+\frac{1}{3!}h^3S'''(0)+\cdots$$

$$S\left(\frac{h}{2}\right)=S(0)+\frac{h}{2}S'(0)+\frac{1}{2!}\left(\frac{h}{2}\right)^2S''(0)+\frac{1}{3!}\left(\frac{h}{2}\right)^3S'''(0)+\cdots$$

如果$S'(0)\neq 0$，那么$S(h), S\left(\frac{h}{2}\right)$逼近$S(0)$的阶为$O(h)$。

若令$S_1(h)=2S\left(\frac{h}{2}\right)-S(h)$，那么当$S''(0)\neq 0$时$S_1(h)$逼近$S(0)$的阶为$O(h^2)$，因此序列$S_1(h), S_1\left(\frac{h}{2}\right), \cdots$就更快地收敛到$S(0)$。同样，还可从$S_1(h)$构造出$S_2(h)$，从$S_2(h)$构造出$S_3(h)$，它们都加速了收敛。这种加速收敛的办法称为外推算法。在数值积分中常用的Richardson（李查逊）外推算法如下。

设一个步长为h的函数F去逼近一个数F^*，其误差估计为

4.4.1 $E(F^*)=F^*-F(h)$

$$=a_1h^{p_1}+a_2h^{p_2}+\cdots+a_kh^{p_k}+\cdots$$

其中

$p_k>p_{k-1}>\cdots>p_2>p_1>0$

a_i，p_i是与h无关的常数，也即是说，F逼近F^*的误差阶是h^{p_1}。如果令

$$F_2(h)=\frac{1}{1-q^{p_1}}[F(qh)-q^{p_1}F(h)], \quad 1-q^{p_1}\neq 0$$

那么$F_2(h)$逼近F^*的误差阶是h^{p_2}。重复这样的做法，可以得到一个算法

4.4.2
$$\begin{cases} F_1(h)=F(h) \\ F_{m+1}(h)=\dfrac{1}{1-q^{p_m}}[F_m(qh)-q^{p_m}F_m(h)] \quad (m=1,2,\cdots) \end{cases}$$

其中q为满足$1-q^{p_m}\neq 0(m=1,2,\cdots)$的适当正数算法。(4.4.2)称为Richardson**外推算法**。

4.4.3 定理 如果F逼近F^*的截断误差由（4.4.1）给出，那么（4.4.2）逼近F^*的误差为

$$F^* - F_{m+1}(h) = a_{m+1}^{(m+1)} h^{p_{m+1}} + a_{m+2}^{(m+1)} h^{p_{m+2}} + \cdots$$

其中$a_k^{(m+1)}$ ($k \geqslant m+1$)为与h无关的常数。

算法（4.4.2）的计算步骤见表（4.4.4），其中ⓘ表示第i步。

4.4.4 表

①$F(h)$				
②$F(qh)$	③$F_2(h)$			
④$F(q^2h)$	⑤$F_2(qh)$	⑥$F_3(h)$		
⑦$F(q^3h)$	⑧$F_2(q^2h)$	⑨$F_3(qh)$	⑩$F_4(h)$	
⋮	⋮	⋮	⋮	⋱

4.4.2 Romberg积分法

定理（4.3.2）给出了复化梯形求积公式的误差估计，为利用Richardson外推算法来加速收敛，可用复化梯形求积公式的误差估计的另一形式。

4.4.5 定理 如果$f^{(2s)}$是$[a,b]$上的连续函数，那么复化梯形求积公式（4.2.1）的误差估计为

$$E_n(f) = \alpha_2 h^2 + \alpha_4 h^4 + \cdots + \alpha_{2s-2} h^{2s-2} + Af^{(2s)}(\eta), \quad \eta \in [a,b]$$

其中

$$\alpha_{2m} = -\frac{B_{2m}}{(2m)!}[f^{(2m-1)}(b) - f^{(2m-1)}(a)] (m=1,2,\cdots,s-1)$$

$$A = -\frac{h^{2s}}{(2s)!}(b-a)B_{2s}$$

此处B_{2k}，$k=1,2,\cdots,s$，为Bernoulli（伯努利）数，表（4.4.6）中给出了部分数值。

4.4.6 表

k	B_k
0	$\frac{1}{1} = 1.00000$
1	$-\frac{1}{2} = -0.50000$
2	$\frac{1}{6} \approx 0.16667$
4	$-\frac{1}{30} \approx -0.03333$
6	$\frac{1}{42} \approx 0.02381$
8	$-\frac{1}{30} \approx -0.03333$
10	$\frac{5}{66} \approx 0.07576$
12	$-\frac{691}{2730} \approx -0.25311$
14	$\frac{7}{6} \approx 1.16667$
16	$-\frac{3617}{510} \approx -7.09216$
18	$\frac{43867}{798} \approx 54.97118$
20	$-\frac{174611}{330} \approx -529.12424$
22	$\frac{854513}{138} \approx 6192.12319$
24	$-\frac{236364091}{2730} \approx -86580.25311$

（当k为≥3的奇数时，$B_k=0$）

在复化梯形求积公式（4.3.1）中把T_n记为$T(h)$，h为步长。

公式（4.3.7）为

$$I(f)\approx T(h)=\frac{h}{2}[f(a)+2\sum_{k=1}^{n-1}f(a+kh)+f(b)]$$

现将步长每次缩小一半，这样可得一序列

$$T(h),T(\frac{h}{2}),T(\frac{h}{2^2}),\cdots$$

显然，这序列收敛到积分值$I(f)$。利用定理（4.4.5）及Richardson外推算法（4.4.2），取$q=\frac{1}{2}$可得到如下算法

$$4.4.7\quad \begin{cases} T_1(h)=T(h) \\ T_{m+1}(h)=\dfrac{T_m\left(\frac{h}{2}\right)-\left(\frac{1}{2}\right)^{2m}T_m(h)}{1-\left(\frac{1}{2}\right)^{2m}} \quad (m=1,2,\cdots) \end{cases}$$

算法（4.4.7）称为Romberg（龙贝格）积分法。$T_{m+1}(h)$逼近$I(f)$的误差估计为

$$4.4.8\quad I(f)-T_{m+1}(h)=a_{m+1}^{(m+1)}h^{2(m+1)}+a_{m+2}^{(m+1)}h^{2(m+2)}+\cdots$$

其中系数$a_{m+1}^{(m+1)},a_{m+2}^{(m+1)}$，…皆与$h$无关。

令

$$T_m^{(k)}=T_m\left(\frac{b-a}{2^k}\right)$$

则计算过程可见表（4.4.9）.表中的每列与对角线都收敛到定积分$I(f)$。

为使用上方便起见，把 Romberg 积分法的计算步骤简述 如下：

1° 计算

$$T_1^{(0)}=\frac{b-a}{2}[f(a)+f(b)]$$

对$l=1,2,\cdots$,计算下列各步:

4.4.9 表

$T_1^{(0)}$					
$T_1^{(1)}$	$T_2^{(0)}$				
$T_1^{(2)}$	$T_2^{(1)}$	$T_3^{(0)}$			
$T_1^{(3)}$	$T_2^{(2)}$	$T_3^{(1)}$	$T_4^{(0)}$		
$T_1^{(4)}$	$T_2^{(3)}$	$T_3^{(2)}$	$T_4^{(1)}$	$T_5^{(0)}$	
$T_1^{(5)}$	$T_2^{(4)}$	$T_3^{(3)}$	$T_4^{(2)}$	$T_5^{(1)}$	$T_6^{(0)}$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$ $\ddots$

2° 计算

$$T_1^{(l)}=\frac{1}{2}\left(T_1^{(l-1)}+\frac{b-a}{2^{l-1}}\sum_{j=1}^{2^{l-1}}f\left(a+(2j-1)\frac{b-a}{2^l}\right)\right)$$

3° 计算表(4.4.6)的第$l+1$行元素

$$T_{m+1}^{(k-1)}=\frac{4^m T_m^{(k)}-T_m^{(k-1)}}{4^m-1}$$

$(m=1,2,\cdots,l,\ k=l,l-1,l-2,\cdots)$

4° 收敛控制:

如果表(4.4.9)的对角线上的最后两相邻元素$T_m^{(0)}$,$T_{m-1}^{(0)}$满足

$|T_m^{(0)}-T_{m-1}^{(0)}|<\varepsilon$(取绝对误差)

或

$$\frac{|T_m^{(0)}-T_{m-1}^{(0)}|}{|T_m^{(0)}|}<\varepsilon \quad \text{(取相对误差)}$$

则以$T_m^{(0)}$作为近似值,否则继续到下面一步.

5° l增加1后转到步骤2°继续做.为使整个过程不会无限做下去,可设$l>l^*$时计算终止,此时达不到要求,计算不成功.

4.4.10 **例** 用Romberg积分法计算$I(f)=\int_0^{\pi}\sin x dx$.

解 计算结果及步骤见表（4.4.11）.

4.4.11 **表**

①0			
②1.57079633	③2.09439511		
④1.89611890	⑤2.00455976	⑥1.99857073	
⑦1.97423160	⑧2.00026917	⑨1.99998313	⑩2.00000555
⑪1.99357034	⑫2.00001659	⑬1.99999975	⑭2.00000001
	⑮1.99999999		
⑯1.99839336	⑰2.00000103	⑱2.00000000	⑲2.00000000
	⑳2.00000000	㉑2.00000000	

可以看到，表的第一列是区间逐步分半的复化求积法所得到的结果，虽然计算简单，但收敛很慢，而使用 Romberg积分则得非常精确的值.

4.5

4.5 Gauss求积公式

积分

4.5.1 $$I(f)=\int_a^b \rho(x)f(x)dx$$

其中积分区间$[a,b]$可以是有限的或无限的，ρ 是$[a,b]$上的权函数(Weight function)，即它满足下而三个条件

1° $\rho(x)\geqslant 0, x\in[a,b]$

2° $\int_a^b \rho(x)dx>0$

3° $\int_a^b |x|^n\rho(x)dx$对所有$n\geqslant 0$可积并有限。

如果取$\rho\equiv 1$，则（4.5.1）就化为通常的积分。

4.5.1 一般理论

现在用n个不等距的节点$x_1,x_2,\cdots,x_n$；$x_k\in[a,b]$，对f进行插值，则有

$$f(x)=\sum_{k=1}^{n}f(x_k)\frac{\omega_n(x)}{(x-x_k)\omega_n'(x_k)}+f[x,x_1,\cdots,x_n]\omega_n(x),\quad x\in[a,b]$$

其中

$\omega_n(x)=(x-x_1)(x-x_2)\cdots(x-x_n)$

$f[x,x_1,\cdots,x_n]$是n阶差商。

用权函数ρ乘上式并在$[a,b]$上积分得

4.5.2 $$I(f)=\int_a^b\rho(x)f(x)dx=\sum_{k=1}^{n}A_kf(x_k)+E(f)$$

其中

4.5.3 $$A_k=\int_a^b\rho(x)\frac{\omega_n(x)}{(x-x_k)\omega_n'(x_k)}dx$$

4.5.4 $$E(f)=I(f)-\sum_{k=1}^{n}A_kf(x_k)=\int_a^b\rho(x)f[x,x_1,x_2,\cdots,x_n]\omega_n(x)dx$$

如果f取为$n-1$次多项式，则有

$$I(f)=\sum_{k=1}^{n}A_kf(x_k)$$

如果f为次数不高于$2n-1$次的多项式，那么f的插值多项式

的余项中$f[x,x_1,\cdots,x_n]$为次数不高于$n-1$次多项式。设$\{g_k\}$为$[a,b]$上关于权函数ρ的正交多项式族（见第3章）。则有

$$f[x,x_1,\cdots,x_n]=\sum_{k=0}^{n-1}c_kg_k(x)$$

因此

$$E(f)=\int_a^b\rho(x)\omega_n(x)\sum_{k=0}^{n-1}c_kg_k(x)dx$$

$$=\sum_{k=0}^{n-1}c_k\int_a^b\rho(x)\omega_n(x)g_k(x)dx$$

如果把非等距节点取成正交多项式g_n的根，那么ω_n和g_n只差一个常数因子，即$\omega_n=\alpha g_n$，这样利用正交性得

$$E(f)=\alpha\sum_{k=0}^{n-1}c_k\int_a^b\rho(x)g_n(x)g_k(x)dx=0$$

所以，只要把插值节点取作在$[a,b]$上带权ρ的正交多项式g_n的根，就可以使具有n个节点的求积公式

4.5.5 $$\int_a^b\rho(x)f(x)dx\approx\sum_{k=1}^{n}A_kf(x_k)$$

的代数精度达到$2n-1$。

4.5.6 **定义** 对于求积公式（4.5.5），A_k由（4.5.3）给定，如果对于次数小于$2n$的多项式f，关系式（4.5.5）为恒等式，则称（4.5.5）为Gauss（高斯）求积公式。

如果节点x_k取为正交多项式g_n的零点，并设a_n为g_n中x^n的系数，$\alpha_n=a_{n+1}/a_n$，$r_n=\int_a^b\rho(x)[g_n(x)]^2dx$，那么（4.5.5)对次数小于$2n$的多项式$f$成为等式，系数$A_k$化为

4.5.7 $A_k = -\dfrac{a_n \gamma_n}{g_n'(x_k)g_{n+1}(x_k)} \quad (k=1,2,\cdots,n)$

4.5.8 **定理** 对于每个$n\geq 1$，存在唯一的求积公式（4.5.5），其中A_k由（4.5.7）给出.对次数小于$2n$的多项式（4.5.5)等式成立.如果$f^{(2n)}$在$[a,b]$上连续,则有

$$\int_a^b \rho(x)f(x)dx = \sum_{k=1}^{n} A_k f(x_k) + \frac{\gamma_n}{a_n^2(2n)!}f^{(2n)}(\eta),$$

$$\eta\in(a,b)$$

由于正交多项式随权函数不同而不同，因此有不同类型的Gauss求积公式.

4.5.2 Gauss-Legendre求积公式

如果在Gauss求积公式（4.5.5）中，权函数取$\rho=1$，积分区间取$[a,b]=[-1,1]$，那么公式（4.5.5）化为

4.5.9 $\int_{-1}^{1} f(x)\,dx \approx \sum_{k=1}^{n} A_k f(x_k)$

由于在$[-1, 1]$上权函数为1的正交多项式是Legendre（勒让德）多项式，所以在公式（4.5.9)中的节点x_k（$k=1,2,\cdots n$)是n次Legendre多项式的零点，而系数为

$$A_k = \frac{-2}{(n+1)P_n'(x_k)P_{n+1}(x_k)} \quad (k=1,2,\cdots,n)$$

如果采用上述的x_k和A_k，则求积公式（4.5.9）被称为**Gauss-Legendre（高斯-勒让德）求积公式.**

Gauss-Legendre求积公式的误差估计

4.5.10 $E_n(f) = \dfrac{2^{2n+1}(n!)^4}{(2n+1)[(2n)!]^2}\cdot\dfrac{1}{(2n)!}f^{(2n)}(\eta)$

$$\eta\in(-1,1)$$

Gauss-Legendre求积公式（4.5.9）中的节点x_k和系数A_k见(4.16)附表(4.1.16)

由(4.16)附表(4.16.1)的节点及系数A_k，利用Gauss-Legendre求积公式很容易计算出积分的近似值，而且精度相当高。表(4.5.11)列出误差估计（4.5.10），可以看出$f^{(2n)}$的系数下降得很快。

4.5.11 表

n	$E_n(f)$	n	$E_n(f)$
1	$\frac{1}{3}f''(\eta)$	5	$\frac{1}{1237732650}f^{(10)}(\eta)$
2	$\frac{1}{135}f^{(4)}(\eta)$	6	$\frac{1}{648984486150}f^{(12)}(\eta)$
3	$\frac{1}{15750}f^{(6)}(\eta)$	7	$\frac{1}{470050192111500}f^{(14)}(\eta)$
4	$\frac{1}{3472875}\cdot f^{(8)}(\eta)$		

4.5.12 例 运用Newton-Cotes求积公式和Gauss-Legendre求积公式计算积分

$$I(f)=\int_{-1}^{1}\sqrt{x+1.5}\,dx$$

解 积分精确值$I(f)=2.399529$，计算结果见表(4.5.12)。

4.5.13 表

n	Gauss-Legendre	Newton-Cotes
2	2.401848	2.288246
3	2.399709	2.395742

由表（4.5.13）可以看出，在节点数目相等的情况下，Gauss-Legendre求积公式的结果更为精确。

对于权函数$\rho=1$的在任意有限区间$[a, b]$上的积分，可以通过自变数的线性变换

$$t=\frac{b-a}{2}x+\frac{a+b}{2}$$

把积分化为标准区间$[-1, 1]$上的积分

$$\int_a^b f(x)dx=\frac{b-a}{2}\int_{-1}^{1} f\left(\frac{a+b+(b-a)x}{2}\right)dx$$

4.5.14 例 用Gauss-Legendre求积公式计算积分

$$I(f)=\int_0^{\pi} e^x\cos(x)dx.$$

解 积分精确值为-12.0703463164。用Gauss-Legendre公式计算的结果 见表(4.5.15)，可以看出这是相当好的(这个积分在例(4.3.5)中曾分别用复化梯形公式和复化Simpson公式计算过)。

4.5.15 表

n	$I_n(f)$	$E_n(f)$
2	-12.33621046570	2.66×10^{-1}
3	-12.12742045017	5.71×10^{-2}
4	-12.07018949029	-1.57×10^{-4}
5	-12.07032853589	-1.78×10^{-5}
6	-12.07034633110	1.47×10^{-8}
7	-12.07034631753	1.14×10^{-9}
8	-12.07034631639	$<5.0\times10^{-12}$

Gauss-Legendre求积公式的误差估计由公式(4.5.10)给出，但是在很多应用中，用被积函数求导的办法来估计其误

差是不方便的，而且有的被积函数是不可微的，因而不能采用这样的办法来估计误差，可以采用下面的两种办法来估计求积公式的误差。

1° 用更高阶的Gauss-Legendre求积公式来检验其结果。

2° 把积分区间分成几个子区间，在这些子区间上采用同样的Gauss-Legendre求积公式。

4.5.16 **例** 用方法1°来计算积分$\int_1^2 \frac{1}{x} dx$。

解 以I_n表示用n个节点的Gauss-Legendre 求积公式计算的结果，有

$I_3 = 0.693122$

$I_4 = 0.693146$

$I_5 = 0.693147$

可以看出，I_4和I_5相当靠近，因此取I_5作为积分的近似值。

应该注意，数值接近有些是虚假的，因此还必须用更高阶公式来检验。例如计算积分

$$I = \int_{-1}^{1} \frac{1}{x^4 + x^2 + 0.9} dx$$

得

$I_3 = 1.585026$

$I_4 = 1.585060$

由此似乎可以看出，取1.585作为近似值就有4位有效数字，但用五个节点的Gauss-Legendre求积公式就看出上面结果不对。积分I的精确值应是1.582233。

方法2°的使用可以取不同的方式。最简单的就是先把积分区间分成二个相等的子区间，而在每个子区间上采用同样的Gauss-Legendre求积公式。

4.5.17 **例** 用方法2°来计算积分

$$I=\int_1^2 \frac{1}{x}dx.$$

解 先把积分区间二等分，

$$I=\int_1^{1.5}\frac{1}{x}dx+\int_{1.5}^2\frac{1}{x}dx$$

$$=\int_{-1}^1\frac{1}{5+u}du+\int_{-1}^1\frac{1}{7+u}du$$

对上面的每个积分用三点Gauss-Legendre求积公式有

$I=0.405464+0.287682=0.693146$

分区间的过程可以继续下去，直到子区间(a_k,b_k)上的积分近似值与$(a_k,\ \frac{1}{2}[a_k+b_k])$及$(\frac{1}{2}[a_k+b_k],b_k)$上的积分近似值之和的差在允许的误差范围之内。

上述计算过程的缺点是前次计算的函数值没有利用，为克服这缺点可采用 Robinson（罗宾森）技巧.其方法如下：先用三点Gauss-Legendre求积公式进行计算，然后把积分区间分为三个子区间，使其每个区间的中点为求积的节点x_k($k=1,2,3$)，于是区间$(-1,\ 1)$分为$(-1,-\alpha)$，$(-\alpha,\alpha)$，$(\alpha,1)$，其中$\alpha=2\sqrt{0.6}-1$。最后把三点Gauss-Legendre求积公式应用到这三个区间。

4.5.18 例 用Robinson技巧计算

$$I=\int_1^2\frac{1}{x}dx.$$

解 $I=\int_{-1}^1\frac{1}{3+u}du$

$$=\int_{-1}^{-\alpha}\frac{1}{3+u}du+\int_{-\alpha}^{\alpha}\frac{1}{3+u}du+\int_{\alpha}^1\frac{1}{3+u}du$$

$$=\beta\int_{-1}^{1}\frac{1}{3-\gamma+\beta v}dv+\alpha\int_{-1}^{1}\frac{1}{3+\alpha v}dv+\beta\int_{-1}^{1}\frac{1}{3+\gamma+\beta v}dv$$

其中：$\alpha=2\sqrt{0.6}-1$，$\gamma=\sqrt{0.6}$，$\beta=1-\gamma$.

计算（三点Gauss-Legendre求积公式）得

$I\approx0.203270+0.370303+0.119574=0.693147$

这样的过程可以继续下去，如果在子区间（a_k,b_k）上直接用三点Gauss-Legendre求积公式，所得到的积分值与在该区间上用Robinson方法的积分值之差在允许误差范围内，即得计算结果.

4.5.3 Gauss-Laguerre求积公式

设积分区间$[a,b]=[0,\infty)$，权函数ρ由公式$\rho(x)=e^{-x}$，$x\in[0,\infty)$给出，那么Gauss求积公式为

$$\int_0^{\infty}e^{-x}f(x)dx\approx\sum_{k=1}^{n}A_kf(x_k) \tag{4.5.19}$$

其中节点$x_k(k=1,2,\cdots,n)$是n次Laguerre（拉盖尔）多项式的根，而系数为

$$A_k=\frac{(n!)^2}{L_n'(x_k)L_{n+1}(x_k)}$$

公式（4.5.18）称为Gauss-Laguerre求积公式，其误差估计为

$$E_n(f)=\frac{(n!)^2}{(2n)!}f^{(2n)}(\eta),\quad \eta\in[0,\infty)$$

节点x_k和系数A_k见(4.16)附表（4.16.2）.

4.5.4 Gauss-Hermite求积公式

如果权函数ρ由表达式$\rho(x)=e^{-x^2}$给出，积分区间$[a,b]=(-\infty,+\infty)$，则称Gauss求积公式

4.5.20 $\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{k=1}^{n} A_k f(x_k)$

为 Gauss-Hermite（高斯—埃尔米特）求积公式，其中节点 $x_k(k=1,2,\cdots,n)$为n次Hermite多项式的根，系数A_k为

$$A_k = \frac{-2^{n+1} n!\sqrt{\pi}}{H_n'(x_k)H_{n+1}(x_k)}$$

Gauss-Hermite求积公式的误差估计为

$$E_n(f) = \frac{n!\sqrt{\pi}}{2^n(2n)!} f^{(2n)}(\eta), \quad \eta \in (-\infty, \infty)$$

节点x_k和系数A_k见(4.16)附表（4.16.3）.

4.5.5 Gauss-Чебышев求积公式

如果权函数ρ由表达式$\rho(x) = \frac{1}{\sqrt{1-x^2}}$给出，积分区间

$[a,b]=[-1,1]$，则称Gauss求积公式

4.5.21 $\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx \approx \sum_{k=1}^{n} A_k f(x_k)$

为Gauss-Чебышев（高斯—切比雪夫）求积 公式，其中节点 x_k是n次Чебышев多项式的根

$$x_k = \cos\frac{(2k-1)\pi}{2n} \quad (k=1,2,\cdots,n)$$

系数为

$$A_k = \frac{\pi}{n}$$

把A_k代入公式（4.5.10）后有

4.5.22 $\int_{-1}^{1} \frac{f(x)}{\sqrt{1-x^2}} dx \approx \frac{\pi}{n} \sum_{k=1}^{n} f(x_k)$

其误差估计为

$$E_n(f)=\frac{2\pi}{2^{2n}(2n)!}f^{(2n)}(\eta),\quad \eta\in[-1,1]$$

4.5.23 **例** 计算积分

$$I(f)=\int_{-1}^{1}\frac{e^x}{\sqrt{1-x^2}}dx$$

使其精确到六位小数.

解 利用Gauss-Чебышев求积公式的误差估计

$$E_n(f)=\frac{2\pi}{2^{2n}(2n)!}e^{\eta},\eta\in(-1,1)$$

有

$$|E_n(f)|\leqslant\frac{2\pi}{2^{2n}(2n)!}e\equiv B_n$$

对于$n=4$，$B_n=1.66\times10^{-6}$；而对于$n=5$，$B_n=4.6\times10^{-9}$.因此取$n=5$，使用公式（4.5.18）得到

$$I(f)\approx\frac{\pi}{5}\sum_{k=1}^{5}\exp\left[\cos\frac{(2k-1)\pi}{10}\right]=3.977463$$

此结果精确到小数点后六位.

4.6 伪-Gauss求积公式

在应用时，有时希望在求积公式中用到被积函数f在几个固定节点上的值.例如,积分区间的端点上的函数$f(x)$的值要求在求积公式中出现，而其余点的选取则要求计算函数值的个数最少.这样的求积公式称为**伪-Gauss求积公式**(Pseudo-Gaussian Quadrature)

4.6.1 Radau求积公式

积分

4.6.1 $I(f)=\int_{-1}^{1} f(x)dx$

的Radau（拉道）求积公式是

4.6.2 $I(f)\approx \frac{2}{n^2}f(-1)+\sum_{k=1}^{n-1} A_k f(x_k)$

其中$x_k(k=1,2,\cdots,n-1)$是多项式ϕ_{n-1}的零点，ϕ_{n-1}由

$$\phi_{n-1}(x)=\frac{1}{1+x}[P_{n-1}(x)+P(x)],\quad x\in[-1,1]$$

给出，式中的P_n是n次Legendre多项式．而权A_k为

4.6.3 $A_k=\frac{1}{1-x_k}\frac{1}{[P'_{n-1}(x_k)]^2}$

$$(x_k\neq -1,k=1,2,\cdots,n-1)$$

求积公式（4.6.2）中仅有一个端点$x=-1$是预先固定的，这个求积公式的误差估计为

4.6.4 $E_n(f)=\frac{2^{2n-1}n[(n-1)!]^4}{[(2n-1)!]^3}f^{(2n-1)}(\eta)$,

$$\eta\in(-1,1)$$

求积公式（4.6.2）中对于 $2\leqslant n\leqslant 5$ 的节点和权见表（4.6.5）．

4.6.5 表

n	x_k $(1\leqslant k\leqslant n)$	A_k $(1\leqslant k\leqslant n)$
2	−1	0.5
	0.333333	1.5
3	−1	0.222222
	−0.289898	0.752806
	0.689898	1.024972

续表

n	$x_k(1\leqslant k\leqslant n)$	$A_k\ (1\leqslant k\leqslant n)$
4	−1	0.125000
	−0.575319	0.657689
	0.181066	0.776387
	0.822824	0.440925
5	−1	0.080000
	−0.720480	0.446207
	−0.167181	0.623653
	00.446314	0.562712
	885792	0.287427

4.6.2 Lobatto求积公式

在伪-Gauss求积公式中很重要的公式就是Lobatto(罗巴托)求积公式

4.6.6 $$I(f)=\int_{-1}^{1}f(x)dx\approx Af(-1)+Af(1)+\sum_{k=1}^{n-2}A_kf(x_k)$$

其中节点x_k是多项式$P'_{n-1}(x)=0$(P_n是n次Lobatto 多项式)的根.

$$A=\frac{2}{n(n-1)}$$

$$A_k=\frac{A}{[P_{n-1}(x_k)]^2}\quad(k=1,2,\cdots,n-2)$$

求积公式(4.6.6)称为 Lobatto 求积公式，也称为 Марков(马尔可夫)求积公式．其误差估计为

$$E_n(f)=-\frac{2^{2n-1}n(n-1)^3[(n-2)!]^4}{(2n-1)[(2n-2)!]^3}f^{(2n-2)}(\eta),$$
$$\eta\in(-1,1)$$

Lobatto求积公式的节点和权见表(4.6.7).

4.6.7 表

n	x_k	A_k
3	0	$\frac{4}{3}$
	±1	$\frac{1}{3}$
4	±0.447214	$\frac{5}{6}$
	±1	$\frac{1}{6}$
5	0	$\frac{32}{45}$
	±0.654654	$\frac{49}{90}$
	±1	$\frac{1}{10}$
6	±0.285232	0.554858
	±0.765055	0.378475
	±1	0.066667

在Lobatto求积中，被积函数f要在$x=\pm1$处计算函数值，因此失去了二个自由度（注意：Radau求积中要在$x=-1$处计算f的值，因而失去一个自由度！）。由此可知，n个节点的Lobatto求积公式仅对$2n-3$次多项式是严格成立的。一般的n个节点的Gauss-Legendre求积公式对$2n-1$次多项式是严格成立的。然而，如果知道被积函数f在区间的端点上取值为0，则宜使用Lobatto求积公式（对Radau求积公式也一样）。这种情况下，$n+2$个节点的Lobatto求积公式只需要计算n个f的值，并且对$2n+1$次多项式是严格成立的。对于n个节点的Gauss-Legendre也需要计算n次f的值，但仅对$2n-1$

次多项式是严格成立。

4.6.8 例 采用5个节点的Lobatto求积公式计算积分

$$I(f)=\int_{-1}^{1}\cos\frac{1}{2}\pi x\ dx$$

$\left(I(f)\text{的精确值为}\frac{4}{\pi}\text{，取5位小数为}1.27324\right)$

解 由于被积函数在积分区间[-1，1]的端点取值为0，因此5个节点的Lobatto求积公式仅需三次函数值的计算。

$$I(f)\approx\frac{49}{90}\cos\frac{1}{2}\pi(-0.654654)+\frac{32}{45}\cos 0$$

$$+\frac{49}{90}\cos\frac{\pi}{2}(0.654654)=1.2732$$

利用Gauss-Legendre三个节点的求积公式，同样要计算三次函数值，其结果是

$I(f)\approx1.27412$

两者相比，显然Lobatto求积公式更为精确。

4.7

4.7 Чебышев求积法

对于积分

$$I(f)=\int_{-1}^{1}f(x)dx,$$

如果f是[-1，1]上的有界变差的连续函数，那么f可以展成一致收敛的Чебышев（切比雪夫）多项式的级数

4.7.1 $f=\frac{1}{2}A_0T_0+A_1T_1+A_2T_2+\cdots$

其中$T_k(k=0,1,\cdots)$是Чебышев多项式，

4.7.2 $A_k=\frac{2}{\pi}\int_{-1}^{1}\frac{f(x)T_k(x)}{\sqrt{1-x^2}}dx=\frac{2}{\pi}\int_0^{\pi}f(\cos\theta)\cos k\theta d\theta$

七

$$(k=0,1,2,\cdots)$$

Чебышев求积法就是先将被积函数f展成（4.7.1），然后取级数的部分和

4.7.3 $$S_n=\frac{1}{2}A_0+A_1T_1+\cdots+A_nT_n$$

来近似f。

但是要直接从（4.7.2)来计算系数，一般是很困难的，为此通常采用S_n的近似式。下面给出两种方法：

1° S_n的近似式取为

4.7.4 $$\widehat{S}_n=\sum_{k=0}^{n}{}' b_rT_r$$

其中求和号Σ的右上方"'"表示第一项要用$\frac{1}{2}$来乘；而

4.7.5 $$b_r=\frac{2}{n+1}\sum_{k=0}^{n}f(x_k)T_r(x_k)$$

上式中x_k取为

4.7.6 $$x_k=\cos\left(\frac{2k+1}{n+1}\ \frac{\pi}{2}\right)\quad(k=0,1,\cdots,n)$$

利用（4.7.4）有

$$I(f)=\int_{-1}^{1}f(x)dx\approx\int_{-1}^{1}\widehat{S}_n(x)\ dx=\sum_{r=0}^{n}{}'\int_{-1}^{1}b_rT_r(x)dx$$

由于

$$\int T_r(x)dx=\begin{cases}T_1(x),\text{当}r=0\\ \frac{1}{4}T_2(x),\ \text{当}r=1\\ \frac{1}{2}\left\{\frac{T_{r+1}(x)}{r+1}-\frac{T_{r-1}(x)}{r-1}\right\},\ \text{当}r>1\end{cases}$$

因此得

八

$$\int_{-1}^{1} T_r(x)\,dx = \begin{cases} 0, & \text{当}r\text{为奇数} \\ \left(\frac{1}{r+1} - \frac{1}{r-1}\right), & \text{当}r\text{为偶数} \end{cases}$$

把上式代入$I(f)$的近似式有

4.7.7 $$I(f) = \int_{-1}^{1} f(x)dx \approx \sum_{r=0}^{[\frac{1}{2}n]} B_{2r+1}$$

其中$[\frac{1}{2}n]$表示$\leqslant \frac{1}{2}n$的最大整数。

$$B_{2s+1} = \frac{b_{2s}}{2s+1}, \quad s = \left[\frac{1}{2}n\right]$$

$$B_{2r+1} = \frac{b_{2r} - b_{2r+2}}{2r+1} \quad (r = 0, 1, \cdots, s-1)$$

2° S_n的另一种近似式取为

4.7.8 $$\overline{S}_n(x) = \sum_{r=0}^{n}{}'' c_r T_r(x)$$

$$= \frac{1}{2}c_0 T_0(x) + c_1 T_1(x) + \cdots + c_{n-1}T_{n-1}(x) + \frac{1}{2}c_n T_n(x)$$

其中求和号Σ右上方“″”表示在和式中的第一项和最后一项用$\frac{1}{2}$来乘。

4.7.9 $$c_r = \frac{2}{n}\sum_{k=0}^{n}{}'' f(x_k)T_r(x_k)$$

式中“″”意义同前，$x_k = \cos\frac{k\pi}{n} \quad (k = 0, 1, \cdots, n)$

求积公式为

4.7.10 $$I(f) = \int_{-1}^{1} f(x)\,dx \approx \int_{-1}^{1} \overline{S}_n(x)dx = \sum_{r=0}^{[\frac{1}{2}n]} C_{2r+1}$$

其中

$$C_{2s+1}=\frac{1}{2s+1}ac_{2s},\quad C_{2s-1}=\frac{1}{2s-1}(c_{2s-2}-ac_{2s})$$

此处

$$s=[\tfrac{1}{2}n],\quad a=\begin{cases}\tfrac{1}{2}, & 当2s=n, n\geqslant 2\\ 1, & 当2s\neq n\end{cases}$$

$$C_{2r+1}=\frac{1}{2r+1}(c_{2r}-c_{2r+2})\quad (r=0,1,\cdots,s-2)$$

4.7.11 例 用Чебышев求积法计算

$$I=\int_1^2\frac{1}{x}dx\quad（精确值为0.69315）$$

解 首先用线性变换$x=\frac{1}{2}(u+3)$把积分区间[1, 2]变为[−1, 1]，这样就有

$$\int_1^2\frac{1}{x}dx=\int_{-1}^1\frac{1}{u+3}du$$

采用方法2°，先取$n=2$，则

$$c_0=\frac{1}{2}f(x_0)T_0(x_0)+f(x_1)T_0(x_1)+\frac{1}{2}f(x_2)T_0(x_2)$$

其中$x_0=-x_2=1$，$x_1=0$

计算得$c_0=\frac{17}{24}$.同理有$c_2=\frac{1}{24}$.于是$C_3=\frac{1}{144}$，$C_1=\frac{11}{16}$

由此得I的近似值

$$I\approx\left(\frac{11}{16}+\frac{1}{144}\right)=\frac{25}{36}=0.69444$$

取$n=4$有：

$c_0=0.70710$

$c_2=0.00674$

$c_4=0.00122$

因此得

$C_5=0.0012$

$C_3=0.00674$

$C_1=0.68627$

这样可以得出I的近似值

$I\approx C_1+C_3+C_5=0.69313$

这是一个很好的近似结果

4.8

4.8 三次样条求积法

用代数插值多项式P_n来代替被积函数 f，可以得到许多实用的求积公式。用三次样条函数S来代替被积函数f，以求得到新的求积公式，这种方法称为三次样条求积（Numerical Integration by Cubic Spline）。

为计算定积分

$$I(f)=\int_a^b f(x)dx$$

可先将区间$[a,b]$分成n等分，其节点为

$$x_k=a+kh \quad (k=0,1,\cdots,n)$$

其中$h=\dfrac{b-a}{n}$，而且在节点的两端处各延拓一点，$x_{-1}=a-h$，$x_{n+1}=a+(n+1)h$。

设S为被积函数f的三次样条插值逼近，那么对任 $x\in[a,b]$有

$$f(x)\approx S(x)=\sum_{k=-1}^{n+1} c_k\Omega_3\left(\frac{x-x_k}{h}\right)$$

由此可以得到

$$I(f)\approx\int_a^b S(x)dx=\sum_{k=-1}^{n+1} c_k\int_a^b \Omega_3\left(\frac{x-x_k}{h}\right)dx \tag{4.8.1}$$

对于$n\geqslant 3$有

$$4.8.2 \quad I(f) \approx \int_a^b S(x)dx = I_n(f)$$

$$= \frac{h}{24}(c_{-1}+c_{n+1}) + \frac{h}{2}(c_0+c_n)$$

$$+ \frac{23}{24}h(c_1+c_{n-1}) + h\sum_{k=2}^{n-2} c_k$$

4.8.1 一般情况的求积公式

为了确定公式(4.8.2)中的 $c_k(k=-1,0,\cdots,n+1)$，可用三次样条的第一插值问题来解决。由三次样条的第一插值问题的条件可以得到如下关系

$$4.8.3 \quad f(x_0) = \sum_{k=-1}^{n+1} c_k\Omega_3(-k) = c_0\Omega_3(0) + c_1\Omega_3(-1) + c_{-1}\Omega_3(1)$$

$$= \frac{2}{3}c_0 + \frac{1}{6}c_1 + \frac{1}{6}c_{-1}$$

$$f(x_n) = \sum_{k=-1}^{n+1} c_k\Omega_3(n-k) = c_n\Omega_3(0) + c_{n+1}\Omega_3(-1)$$

$$+ c_{n-1}\Omega_3(1) = \frac{2}{3}c_n + \frac{1}{6}c_{n+1} + \frac{1}{6}c_{n-1}$$

$$f'(x_0) = \frac{1}{h}\sum_{k=-1}^{n+1} c_k\Omega_3'(-k) = \frac{c_1 - c_{-1}}{2h}$$

$$f'(x_n) = \frac{1}{h}\sum_{k=-1}^{n+1} c_k\Omega_3'(n-k) = \frac{c_{n+1} - c_{n-1}}{2h}$$

$$f(x_k) = \sum_{j=-1}^{n+1} c_j\Omega_3(k-j) = c_k\Omega_3(0) + c_{k+1}\Omega_3(-1)$$

$$+ c_{k-1}\Omega_3(1) = \frac{2}{3}c_k + \frac{1}{6}c_{k+1} + \frac{1}{6}c_{k-1}$$

$$(k=1,2,\cdots,n-1)$$

由公式（4.8.3）的各式可以得到

$$h\sum_{k=1}^{n-1} f(x_k)=h\left(\sum_{k=2}^{n-2} c_k\right)+\frac{h}{6}(c_0+c_n)$$

$$+\frac{5}{6}h(c_1+c_{n-1})-\frac{h}{2}[f(x_0)+f(x_n)]$$

$$=\frac{h}{3}(c_0+c_n)+\frac{h}{12}(c_1+c_{n-1})+\frac{h}{12}(c_{-1}+c_{n+1})$$

由上面两式得

$$\frac{1}{2}h\left[f(x_0)+f(x_n)+2\sum_{k=1}^{n-1} f(x_k)\right]$$

$$=I_n(f)+\frac{h}{24}(c_{-1}-c_1+c_{n+1}-c_{n-1})$$

其中$I_n(f)$由（4.8.2）所定义．再利用公式（4.8.2)的第三式和第四式可得到求积公式

4.8.4 $I(f)\approx I_n(f)$

$$=\frac{h}{2}\left(f(x_0)+f(x_n)+2\sum_{k=1}^{n-1} f(x_k)\right)+\frac{h^2}{12}\left(f'(x_0)-f'(x_n)\right)$$

这就是三次样条第一插值问题的求积公式．这个求积公式的误差估计为

4.8.5 $E_n(f)=I(f)-I_n(f)=\frac{b-a}{720}h^4 f^{(4)}(\eta),\quad \eta\in(a,b)$

4.8.2 简单情况的求积公式

公式（4.8.2）仅对$n\geqslant 3$时才成立，因此对于$n=1$，$n=2$这两种特殊情况必须单独求出．对于$n=1$：

$$\int_a^b \Omega_3\left(\frac{x-x_0}{h}\right)dx=\int_a^b \Omega_3\left(\frac{x-x_1}{h}\right)dx=h\int_0^1 \Omega_3(x)dx=\frac{11}{24}h$$

$$\int_a^b \Omega_3\left(\frac{x-x_{-1}}{h}\right)dx=\int_a^b \Omega_3\left(\frac{x-x_2}{h}\right)dx=h\int_1^2 \Omega_3(x)\ dx=\frac{1}{24}h$$

由（4.8.1）得

$$I(f)\approx\frac{h}{24}(c_{-1}+c_2)+\frac{11}{24}h(c_0+c_1)$$

对于三次样条第一插值问题，$c_k(k=-1,0,1,2)$ 可由以下方程组确定

$$\begin{cases} -c_{-1}+c_1=2hf'(x_0) \\ c_{-1}+4c_0+c_1=6f(x_0) \\ c_0+4c_1+c_2=6f(x_1) \\ -c_0+c_2=2hf'(x_1) \end{cases}$$

解之，得

$$\begin{cases} c_{-1}=(2f(x_1)-f(x_0))-\dfrac{h}{3}(2f'(x_1)+7f'(x_0)) \\ c_0=(2f(x_0)-f(x_1))+\dfrac{h}{3}(2f'(x_0)+f'(x_1)) \\ c_1=(2f(x_1)-f(x_0))-\dfrac{h}{3}(2f'(x_1)+f'(x_0)) \\ c_2=(2f(x_0)-f(x_1))+\dfrac{h}{3}(2f'(x_0)+7f'(x_1)) \end{cases}$$

由此得到

4.8.6 $$\int_a^b (fx)dx\approx\frac{h}{2}(f(x_0)+f(x_1))+\frac{h^2}{12}(f'(x_0)-f'(x_1))$$

与$n=1$的推导相似，可以得到$n=2$的求积公式

4.8.7 $$\int_a^b f(x)dx$$

$$\approx\frac{1}{2}h(f(x_0)+2f(x_1)+f(x_2))\quad+\frac{1}{12}h^2(f'(x_0)-f'(x_2))$$

4.8.8 例 用三次样条求积法计算$\int_{0.5}^{1}\sqrt{x}\,dx$（精确值为0.4309644）.

解 对于$n=1$，利用公式（4.8.6）有

$$\int_{0.5}^{1}\sqrt{x}\,dx\approx\frac{0.5}{2}[\sqrt{0.5}+1]+\frac{0.5^2}{12}\left[\frac{1}{2\sqrt{0.5}}-\frac{1}{2}\right]$$

$$=0.43109142$$

对于$n=2$，利用公式（4.8.7）有

$$\int_{0.5}^{1}\sqrt{x}\,dx\approx0.43097338$$

对于$n=3$，利用公式（4.8.4）有

$$\int_{0.5}^{1}\sqrt{x}\,dx\approx0.43096623$$

可以看出，三次样条求积公式对于$n=1,2,3$分别有三位、四位、五位有效数字.

4.9

4.9 自适应积分法

被积函数在整个积分区间$[a,b]$上变化不一定是均匀的，如在某一点附近函数变化非常急剧，而在其余地方的变化就比较平缓.为了使计算达到预定的精度又要省工作量,则可以在函数变化急剧的部分增多节点，即子区间分得细，而在函数变化平缓的地方减少节点.这个方法就是**自适应积分法**(Adaptive Numerical Integation).

4.9.1 自适应Simpson方法

采用逐次将区间二等分的办法.为写法统一,将积分区间$[a,b]$记为$[a,a+h]$,其中$h=b-a$为区间长度，称原区间为0级区间.在区间$[a,h]$上使用Simpson公式（4.2.13），把结

果记作

4.9.1 $S^{(1)}_{a,a+h}=\frac{h}{6}\left[f(a)+4f\left(a+\frac{h}{2}\right)+f(a+h)\right]$

将区间分成二个相等的子区间$[a,a+\frac{h}{2}]$和$[a+\frac{h}{2},a+h]$，

这二个子区间称为1级子区间，其长度为 $\frac{h}{2}$。在每个1级子

区间上采用Simpson公式计算积分，然后相加并令

4.9.2 $S^{(2)}_{a,a+h}=S^{(1)}_{a,a+\frac{h}{2}}+S^{(1)}_{a+\frac{h}{2},a+h}$

再将1级子区间中的一个或所有的两个二等分，所得的子区间

称为2级子区间，其长度为$\frac{1}{2^2}h,\cdots$，如此继续下去，最后

将区间$[a,a+h]$分成n个子区间$[a_i,a_{i+1}](i=0,1,\cdots,n-1)$。
这样有

4.9.3 $a=a_0<a_1<\cdots<a_i<a_{i+1}<\cdots<a_n=b=a+h$

子区间的长度一般是不同的，如果子区间$[a_i,a_{i+1}]$是r级，则其长度为

$$a_{i+1}-a_i=\frac{h}{2^r}$$

实际上，区间的划分（4.9.3）是根据函数的变化情况而定的。函数变化平缓的地方，子区间大，函数变化急剧的地方，子区间就小。

设$S_{a,a+h}$表示$I(f)=\int_a^b f(x)dx$的近似值，那么有

$$S_{a,a+h}=\sum_{i=0}^{n-1}S^{(2)}_{a_i,a_{i+1}}$$

若计算$I(f)$的允许误差为ε，则需有

$$|S_{a,a+h}-I(f)|<\varepsilon$$

如果记$I_{\alpha,\beta}(f)=\int_\alpha^\beta f(x)dx$，则上式化为

4.9.4 $$\left|\sum_{i=0}^{n-1}\left(S^{(2)}_{a_i,a_{i+1}}-I_{a_i,a_{i+1}}(f)\right)\right|<\varepsilon$$

如果取每个r级子区间上误差控制为$\frac{1}{2^r}\varepsilon$，即取$[a_i,a_{i+1}]$为r级子区间，则当

4.9.5 $$|S^{(2)}_{a_i,a_{i+1}}-I_{a_i,a_{i+1}}(f)|<\frac{1}{2^r}\varepsilon$$

时，(4.9.4) 可以得到满足.

要直接验证 (4.9.5) 是否成立是相当困难的，可以采用间接方法.设f在$[a,a_{i+1}]$上为五次连续可微，则Simpson求积公式的误差

$$I_{a_i,a_{i+1}}(f)-S^{(1)}_{a_i,a_{i+1}}=-\frac{(a_{i+1}-a_i)^5}{90}f^{(4)}(\eta_1)$$

$$=-\frac{1}{90}(a_{i+1}-a_i)^5f^{(4)}\left(a_i+\frac{a_{i+1}-a_i}{2}\right)+o((a_{i+1}-a_i)^6)$$

$$a_i<\eta_1<a_{i+1}$$

$$I_{a_i,a_{i+1}}(f)-S^{(2)}_{a_i,a_{i+1}}=-\frac{1}{1440}(a_{i+1}-a_i)^5f^{(4)}(\eta_2)$$

$$=-\frac{1}{1440}(a_{i+1}-a_i)^5f^{(4)}\left(a_i+\frac{a_{i+1}-a_i}{2}\right)+o((a_{i+1}-a_i)^6)$$

$$a_i<\eta_2<a_{i+1}$$

如果忽略$o((a_{i+1}-a_i)^6)$，则有

$$|I_{a_i,a_{i+1}}(f)-S^{(2)}_{a_i,a_{i+1}}|=\frac{1}{15}|S^{(2)}_{a_i,a_{i+1}}-S^{(1)}_{a_i,a_{i+1}}|$$

因此，如果

4.9.6 $$|S^{(2)}_{a_i,a_{i+1}}-S^{(1)}_{a_i,a_{i+1}}|<15\varepsilon\frac{1}{2^r}$$

那么 (4.9.5) 就满足，从而 (4.9.4) 也满足.

在逐次二等分区间的过程中，可以根据不等式(4.9.6)判断是否要将一个r级子区间继续分成二个相等的$r+1$级子区间.如果(4.9.6)成立，则认为在子区间$[a_i, a_{i+1}]$上已达到计算的精确度，因而可以再考虑与$[a_i, a_{i+1}]$右相邻的那个子区间；否则，将继续分$[a_i, a_{i+1}]$为二个相等的$r+1$级子区间。

在实际计算中，(4.9.6)的右边经常用$10\varepsilon\frac{1}{2^r}$来代替。

4.9.2 计算步骤

将区间$[a, a+h]$，$h=b-a$分成二个相等的1级子区间$\left[a, a+\frac{h}{2}\right]$和$\left[a+\frac{h}{2}, a+h\right]$，区间长度为$\frac{h}{2}$。在这二个1级子区间上使用Simpson求积公式(4.2.13)，得到结果$S^{(1)}_{a, a+\frac{h}{2}}$和$S^{(1)}_{a+\frac{h}{2}, a+h}$。然后在1级区间上计算

$$S^{(2)}_{a, a+\frac{h}{2}} = S^{(1)}_{a, a+\frac{h}{2^2}} + S^{(1)}_{a+\frac{h}{2^2}, a+\frac{h}{2}}$$

在1级区间$\left[a, a+\frac{h}{2}\right]$上比较$S^{(1)}_{a, a+\frac{h}{2}}$与$S^{(2)}_{a, a+\frac{h}{2}}$，如果不等式

$$\left| S^{(1)}_{a, a+\frac{h}{2}} - S^{(2)}_{a, a+\frac{h}{2}} \right| < 10\frac{\varepsilon}{2} \tag{4.9.7}$$

成立，则说明在1级子区间$\left[a, a+\frac{h}{2}\right]$上已达到要求，然后在下一个1级子区间$\left[a+\frac{h}{2}, a+h\right]$上计算

$$S^{(2)}_{a+\frac{h}{2}, a+h} = S^{(1)}_{a+\frac{h}{2}, a+\frac{3}{2^2}h} + S^{(1)}_{a+\frac{3}{2^2}h, a+h}$$

如果不等式

$$\left|S^{(2)}_{a+\frac{h}{2},\,a+h} - S^{(1)}_{a+\frac{h}{2},\,a+h}\right| < 10\frac{\varepsilon}{2} \tag{4.9.8}$$

成立，则认为在整个区间$[a,\ a+h]$上计算完成，令

$S_{a,\,a+h} = S^{(2)}_{a,\,a+\frac{h}{2}} + S^{(2)}_{a+\frac{h}{2},\,a+h}$

这就是 $I(f)$的近似值.

如果不等式（4.9.7）和（4.9.8）中有一个不成立，例如（4.9.7）不成立，则不考虑 1 级子区间$\left[a+\frac{h}{2},a+h\right]$，而将$\left[a,\ a+\frac{h}{2}\right]$分成二个相等的 2 级子区间$\left[a,\ a+\frac{h}{2^2}\right]$和$\left[a+\frac{h}{2^2},\ a+\frac{h}{2}\right]$.对 2 级子区间$\left[a,a+\frac{h}{2^2}\right]$采用Simpson公式（4.2.13)计算 $S^{(1)}_{a,\,a+\frac{1}{2^2}h}$和$S^{(2)}_{a,\,a+\frac{1}{2^2}h} = S^{(1)}_{a,\,a+\frac{1}{2^3}h} + S^{(1)}_{a+\frac{1}{2^3}h,\,a+\frac{1}{2^2}h}$. 然后比较 $S^{(1)}_{a,\,a+\frac{1}{2^2}h}$和$S^{(2)}_{a,\,a+\frac{1}{2^2}h}$，如果不等式

$$\left|S^{(1)}_{a,\,a+\frac{1}{2^2}h} - S^{(2)}_{a,\,a+\frac{1}{2^2}h}\right| < 10\frac{\varepsilon}{2^2} \tag{4.9.9}$$

满足，则说明在 2 级子区间$\left[a,\ a+\frac{1}{2^2}h\right]$上计算已达到要求，继续在其右边 2 级子区间$\left[a+\frac{1}{2^2}h,\ a+\frac{1}{2}h\right]$上进行类似计算，并满足类似于(4.9.9)的不等式，此时可取

$$S_{a,\,a+h} = S^{(2)}_{a,\,a+\frac{1}{2^2}h} + S^{(2)}_{a+\frac{1}{2^2}h,\,a+\frac{1}{2}h} + S^{(2)}_{a+\frac{1}{2}h,\,a+h}$$

为 $I(f)$的近似值.

如果不等式（4.9.9）不成立，则再进一步将 2 级子区

间$\left[a,\ a+\frac{h}{2^2}\right]$分成二个相等的3级子区间，等等，类似上述步骤继续进行。

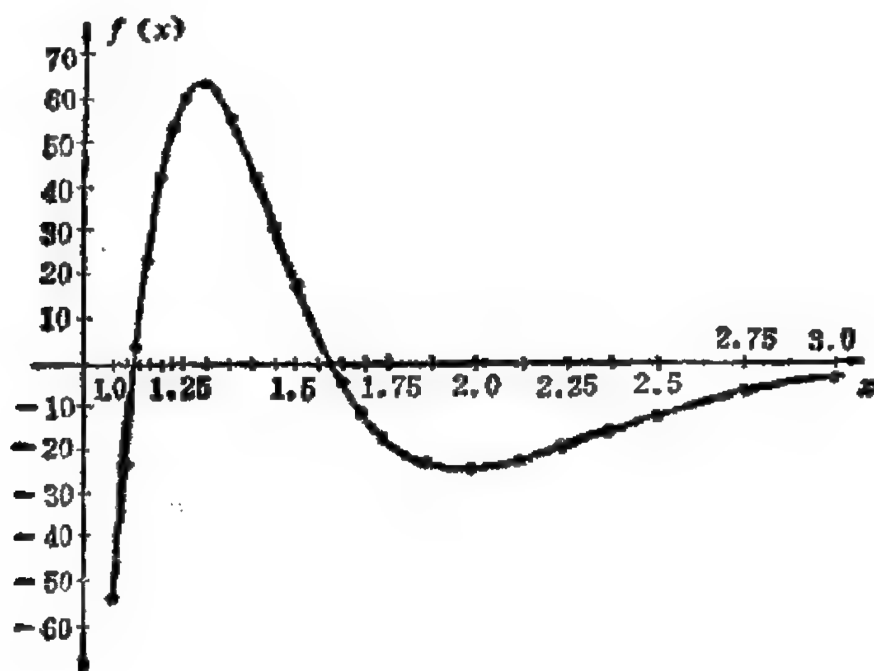
4.9.10 例 公式$f(x)=\frac{100}{x^2}\sin\frac{10}{x}$，$x\in[1,3]$给出函数$f$的图象如图（4.9.11）所示。用自适应Simpson公式计算$\int_1^3 f(x)dx$，使其容许误差为10^{-4}。

解 采用自适应Simpson公式，在23个子区间（其节点分布见图4.9.11）上用$n=2$的复化Simpson公式，得数值结果为-1.426014，其精度达到1.4×10^{-6}，在整个过程中计算函数值的次数为93。

如果采用128个小区间复合Simpson求积公式，其数值结果为-1.426059，与准确值之差为2.4×10^{-6}，此处计算函数值的次数为257。

计算函数值是费时间的，因此在相同的精度内自适应Simpson积分法比复化Simpson公式更为优越。

4.9.11 图



4.10 奇异积分

奇异积分 (Singular Integral) 一般不能用前面所述方法来进行计算，因此必须针对具体积分选择适当的方法.

4.10.1 积分变量替换

采用积分变量替换(Change of the Variable of Integration) 可以消除奇异性.

4.10.1 例 计算积分

$$I=\int_0^b \frac{f(x)}{\sqrt{x}}dx$$

其中 f 是一个充分光滑的函数。

解 为消去奇异性，令

$x=u^2$, $0\leqslant u\leqslant\sqrt{b}$

积分化为

$$I=2\int_0^{\sqrt{b}} f(u^2)du$$

这个积分的被积函数是光滑的，因此可以应用标准的求积方法.

4.10.2 例 考虑积分

$$I=\int_0^1 \sin(x)\sqrt{1-x^2}\,dx$$

解 被积函数的第二个因子的一阶导数在$x=0$处有奇点。采用变换

$u=\sqrt{1-x}$

此时

$$I=2\int_0^1 u^2\sqrt{2-u^2}\sin(1-u^2)du$$

这个积分的被积函数是可微的，因此可以应用通常的方法进行计算.

4.10.2 奇异性的解析处理

奇异性的解析处理 (Analytic Treatment of Singularity) 也称区间的截去方法，是把积分区间分成两部分，使一部分有奇异点而另一部分没有奇异点. 如果

$$I=\int_a^b f(x)dx$$

中被积函数 f 在 $x=a$ 处有奇异点，则适当地选取小数 $\delta>0$，可使在小区间 $[a,a+\delta]$ 上的积分值处在允许的误差范围之内，即

$$\left|\int_a^{a+\delta} f(x)dx\right|<\varepsilon$$

而对于积分

$$\int_{a+\delta}^b f(x)dx$$

则可以按标准的数值积分方法进行.

4.10.3 例 计算积分

$$\int_0^1 \frac{g(x)}{x^{\frac{1}{2}}+x^{\frac{1}{3}}}dx$$

其中 g 在$[0, 1]$上充分光滑，且满足 $|g(x)|\leqslant 1, x\in[0,1]$。

解 因为在$[0, 1]$上，$x^{\frac{1}{2}}\leqslant x^{\frac{1}{3}}$，则有

$$\left|\frac{g(x)}{x^{\frac{1}{2}}+x^{\frac{1}{3}}}\right|\leqslant\frac{1}{2x^{\frac{1}{2}}}$$

因此

$$\left|\int_0^\delta \frac{g(x)}{x^{\frac{1}{2}}+x^{\frac{1}{3}}}dx\right|\leqslant\frac{1}{2}\int_0^\delta \frac{1}{x^{\frac{1}{2}}}dx=\delta^{\frac{1}{2}}$$

如果精度要求为 10^{-8}，则 $\delta \leqslant 10^{-6}$ 而在$[\delta,1]$上的积分

$$\int_{\delta}^{1} \frac{g(x)}{x^{\frac{1}{2}}+x^{\frac{1}{3}}} dx$$

可以采用标准方法进行计算。

4.10.4 例 计算积分

$$I=\int_{0}^{b} f(x) \ln x dx$$

解 先分成两部分，

$$I=\int_{0}^{\delta} f(x) \ln x dx+\int_{\delta}^{b} f(x) \ln x dx \equiv I_{1}+I_{2}$$

假定在$[\delta,b]$上 f 充分光滑，则可用标准方法计算 I_2，并假设在$[0,\varepsilon]$上 f 可展成收敛的 Taylor 级数，那么

$$I_{1}=\int_{0}^{\delta} f(x) \ln x dx=\int_{0}^{\delta}\left(\sum_{j=0}^{\infty} a_{j} x^{j}\right) \ln x dx$$

$$=\sum_{j=0}^{\infty} a_{j} \frac{\delta^{j+1}}{j+1}\left(\ln \delta-\frac{1}{j+1}\right)$$

对于给定的精度，可以对级数进行估计。取 $f(x)=\cos x$，$b=4\pi$，即计算

$$I=\int_{0}^{4\pi} \cos x \ln x dx$$

取 $\delta=0.1$，则

$$I_{1}=\int_{0}^{0.1} \cos x \ln x dx$$

$$=\delta(\ln \delta-1)-\frac{\delta^{3}}{6}\left(\ln \delta-\frac{1}{3}\right)+\frac{\delta^{5}}{600}\left(\ln \delta-\frac{1}{5}\right) \cdots \cdots$$

这是一个交错级数，取前三项可以求得 I_1 的相当精确的值，而在$[0.1,4\pi]$上的积分 I_2 可以用标准的方法求出。

4.10.3 乘积积分

考虑积分

$$I(f)=\int_a^b w(x)f(x)dx$$

其中 w 是一个奇异的权函数，f 是一个光滑的函数。构造一个函数序列 f_n，使得

1° 当 $n\to\infty$ 时有

$$\|f-f_n\|_\infty=\max_{a\leqslant x\leqslant b}|f(x)-f_n(x)|\to 0$$

2° 积分

$$I_n(f)=\int_a^b w(x)f_n(x)dx$$

容易计算。

乘积积分方法（Product Integration Method）通常采用 f 的分段多项式插值 f_n 来确定 $I_n(f)$，可以用乘积梯形方法(Product Trapezoidal Method)来计算积分

4.10.5 $$I(f)=\int_0^b f(x)\ln x dx$$

令 $n\geqslant 1$，$h=\frac{b}{n}$，$x_j=jh$ $(j=0,1,\cdots,n)$，f_n 定义为 f 在节点 x_0，x_1，…，x_n 上的分段线性函数插值。对于 $j=1$，2，3，…，n，定义

4.10.6 $$f_n(x)=\frac{1}{h}[(x_j-x)f(x_{j-1})+(x-x_{j-1})f(x_j)],\quad x_{j-1}\leqslant x\leqslant x_j$$

如果 f 在 $[a, b]$ 上二次连续可微，那么利用插值多项式的误差估计可以得到

$$\|f-f_n\|_\infty\leqslant\frac{h^2}{8}\|f''\|_\infty$$

由此有

$$|I(f)-I_n(f)|\leqslant \frac{h^2}{8}\|f''\|_\infty\int_0^b |\ln x|\,dx$$

而 $I_n(f)$ 是容易计算的，

4.10.7 $I_n(f)$

$$=\sum_{j=1}^{n}\int_{x_{j-1}}^{x_j}(\ln x)\left[\frac{(x_j-x)f(x_{j-1})+(x-x_{j-1})f(x_j)}{h}\right]dx$$

$$=\sum_{k=0}^{n}A_k f(x_k)$$

其中

$$\begin{cases} A_0=\dfrac{1}{h}\displaystyle\int_{x_0}^{x_1}(x_1-x)\ln x\,dx \\ A_n=\dfrac{1}{h}\displaystyle\int_{x_{n-1}}^{x_n}(x-x_{n-1})\ln x\,dx \\ A_j=\dfrac{1}{h}\displaystyle\int_{x_{j-1}}^{x_j}(x-x_{j-1})\ln x\,dx+\dfrac{1}{h}\displaystyle\int_{x_j}^{x_{j+1}}(x_{j+1}-x)\ln x\,dx \end{cases}$$

$$(j=1,\cdots,n-1)$$

作变数的替换 $x-x_{j-1}=uh$，$0\leqslant u\leqslant 1$，有

$$\frac{1}{h}\int_{x_{j-1}}^{x_j}(x-x_{j-1})\ln x\,dx=\frac{h}{2}\ln h+h\int_0^1 u\ln(j-1+u)\,du$$

和

$$\frac{1}{h}\int_{x_{j-1}}^{x_j}(x_j-x)\ln x\,dx=\frac{h}{2}\ln h+h\int_0^1(1-u)\ln(j-1+u)\,du$$

令

4.10.8 $\psi_1(k)=\int_0^1 u\ln(u+k)\,du$

$$\psi_2(k)=\int_0^1 (1-u)\ln(u+k)du \qquad (k=0, 1,\cdots,n)$$

可得

4.10.9 $$w_0=\frac{h}{2}\ln h+h\psi_2(0)$$

$$w_n=\frac{h}{2}\ln h+h\psi_1(n-1)$$

$$w_j=h\ln h+h[\psi_1(j-1)+\psi_2(j)]$$

$$(j=1,2,\cdots,n-1)$$

$\psi_1(k)$ 和 $\psi_2(k)$ 不依赖于 h, b 或 n，因此它们可以预先计算好。$\psi_1(k)$，$\psi_2(k)$ 的前 8 个数值见表（4.10.10）。

4.10.10 表

k	$\psi_1(k)$	$\psi_2(k)$
0	−0.250	−0.75
1	0.250	0.1362943611
2	0.4883759281	0.4211666768
3	0.6485778545	0.6007627239
4	0.7695705457	0.7324416720
5	0.8668602747	0.8365069785
6	0.9482428376	0.9225713904
7	1.018201652	0.9959596385

4.10.4 Канторович 方法

积分

$$I(f)=\int_a^b f(x)dx$$

的被积函数 f 存在一个奇点，Канторович（康托洛维奇）方法不是直接对积分 $I(f)$ 进行求积，而是选取一个函数 g，使得它与 f 有相同的奇点，并在给定的积分区间 $[a, b]$ 上可

以积出，而且 $f-g$ 有一定阶的导数．把积分写成

$$\int_a^b f(x)dx=\int_a^b g(x)dx+\int_a^b [f(x)-g(x)]dx$$

右边的第一个积分可以直接积出，而第二个积分可以应用标准的数值求积公式计算出来．

函数 g 的选取，有很多方法．如，设被积函数 f 用公式

4.10.11 $f(x)=(x-c)^{\alpha}\varphi(x)$，$a\leqslant c\leqslant b$，$x\in[a,b]$

来表示，其中 $-1<\alpha<0$，φ 在$[a, b]$上足够光滑．φ在$x=c$处展成 Taylor 级数，则可以得到

4.10.12

$$f(x)=\left[\varphi(c)(x-c)^{\alpha}+\frac{\varphi'(c)}{1!}(x-c)^{\alpha+1}+\frac{\varphi''(c)}{2!}(x-c)^{\alpha+2}+\cdots\cdots+\frac{\varphi^{(k)}(c)}{k!}(x-c)^{\alpha+k}\right]$$

$$+(x-c)^{\alpha}\left[\varphi(x)-\varphi(c)-\frac{\varphi'(c)}{1!}(x-c)-\frac{\varphi''(c)}{2!}(x-c)^2-\cdots-\frac{\varphi^{(k)}(c)}{k!}(x-c)^k\right]$$

上式右边第一个方括号中的是一个幂函数，可以逐项求其积分；而第二个方括号内已无奇点，且相当光滑，可以用标准的数值求积公式计算出来．

4.10.13 例 计算

$$I=\int_0^{\frac{1}{2}}\frac{1}{\sqrt{x(1-x)}}dx$$

的近似值．

解 被积函数在 $x=0$ 处间断，且可用公式

$$f(x)=x^{-\frac{1}{2}}(1-x)^{-\frac{1}{2}}$$

表示，于是$a=-\frac{1}{2}$，$c=0$，$\varphi(x)=(1-x)^{-\frac{1}{2}}$。由 Taylor 级数展开

$$\varphi(x)=1+\frac{1}{2}x+\frac{3}{8}x^2+\frac{5}{16}x^3+\frac{35}{128}x^4+R_4(x)$$

因此 f 可以写成

$$f(x)=\left[x^{-\frac{1}{2}}+\frac{1}{2}x^{\frac{1}{2}}+\frac{3}{8}x^{\frac{3}{2}}+\frac{5}{16}x^{\frac{5}{2}}+\frac{35}{128}x^{\frac{7}{2}}\right]+\frac{\psi(x)}{\sqrt{x}}$$

其中

$$\psi(x)=\frac{1}{\sqrt{1-x}}-\left(1+\frac{1}{2}x+\frac{3}{8}x^2+\frac{5}{16}x^3+\frac{35}{128}x^4\right)$$

因此，

$$I=\int_0^{\frac{1}{2}}\left(x^{-\frac{1}{2}}+\frac{1}{2}x^{\frac{1}{2}}+\frac{3}{8}x^{\frac{3}{2}}+\frac{5}{16}x^{\frac{5}{2}}+\frac{35}{128}x^{\frac{7}{2}}\right)dx+I_1$$

$$=1.5691585+I_1$$

其中

$$I_1=\int_0^{\frac{1}{2}}\frac{\psi(x)}{\sqrt{x}}dx$$

用 $n=10$ 的复化 Simpson 公式计算 I_1，得 $I_1=0.0016385$，由此得 $I=1.5707970$。

为比较起见，求出 I 的精确值，$I=\frac{\pi}{2}=1.5707963$。由此可见是相当精确的。

4.10.5 Gauss 求积

如果被积函数分解成二个函数的乘积，那么奇异积分常可应用 Gauss 型求积，考虑

4.10.14 $I(f)=\int_a^b w(x)f(x)dx$

其中 w 是一个固定的非负的权函数，它在积分区间 $[a, b]$ 上存在一个或几个奇异点，并且积分 $\int_a^b w(x)x^k dx$ （$k=0, 1, \cdots, n$）存在，而 f 是一个充分光滑的函数。这样可以按（4.5）的方法来确定积分公式的节点和系数。w 有以下几种特殊情况：

1° $w(x)=(1-x^2)^{\frac{1}{2}}$，相应的求积公式

4.10.15 $\int_{-1}^{1}\sqrt{1-x^2}\,f(x)dx\approx\sum_{k=1}^{n}A_k f(x_k)$

其中

$$x_k=\cos\frac{k+1}{n+1}\pi,\quad A_k=\frac{\pi}{n+1}\sin^2\frac{k+1}{n+1}\pi$$

误差估计为

$$E_n=\frac{\pi}{(2n)!\,2^{2n+1}}f^{(2n)}(\eta),\quad \eta\in(-1,1)$$

2° $w(x)=\sqrt{x}\,\frac{1}{\sqrt{1-x}}$，相应的求积公式

4.10.16 $\int_0^1\sqrt{\frac{x}{1-x}}f(x)dx\approx\sum_{k=1}^{n}A_k f(x_k)$

其中

$$x_k=\cos^2\frac{2k-1}{2n+1}\frac{\pi}{2},\quad A_k=\frac{2\pi}{2n+1}x_k$$

误差估计

$$E_n(f)=\frac{\pi}{(2n)!2^{4n+1}}f^{(2n)}(\eta),\quad \eta\in(0,1)$$

3° $w(x)=(1-x)^{\frac{1}{2}}$，相应的求积公式

4.10.17 $$\int_0^1\sqrt{1-x}f(x)dx\approx\sum_{k=1}^{n}A_kf(x_k)$$

其中

$x_k=1-z_k^2$

（z_k 是 $2n+1$ 次Legendre 多项式 P_{2n+1} 的第 k 个正零点）

$A_k=2z_k^2A_k^{(2n+1)}$

此处$A_k^{(2n+1)}=\dfrac{2}{(1-z_k^2)[P'_{2n+1}(z_k)]^2}$ 是 $2n+1$ 个节点 Gauss -Legendre 求积公式中相应于节点 z_k 的系数。

误差估计

$$E_n(f)=\frac{2^{4n+3}[(2n+1)!]^4}{(2n)!(4n+3)[(4n+2)!]^2}f^{(2n)}(\eta),\quad \eta\in(0,1)$$

4° $w(x)=(1-x)^{-\frac{1}{2}}$，相应的求积公式

4.10.18 $$\int_0^1\frac{f(x)}{\sqrt{1-x}}dx\approx\sum_{k=1}^{n}A_kf(x_k)$$

其中$x_k=1-z_k^2$（z_k是$2n$次Legendre多项式$P_{2n}(x)$的第k个正零点），

$A_k=2A_k^{(2n)}$

此处 $A_k^{(2n)}=\dfrac{2}{(1-z_k^2)[P'_{2n}(z_k)]^2}$ 是 $2n$ 个节点的 Gauss -Legendre 求积公式中相应于节点 z_k 的系数。

误差估计

$$E_n(f)=\frac{2^{4n+1}[(2n)!]^3}{(4n+1)[(4n)!]^2}f^{(2n)}(\eta),\quad \eta\in(0,1)$$

5° $w(x)=(1-x^2)^{-\frac{1}{2}}$，相应的求积公式为 Gauss-Чебышев公式（见 4.5.21）.

4.11

4.11 无穷区间上的积分

对于无穷区间上的积分

4.11.1 $$I(f)=\int_a^\infty f(x)dx$$

除可用 Gauss-Laguerre 求积公式和 Gauss-Hermite求积公式（见 4.5）来计算外，还有如下其它计算方法.

4.11.1 替换变量

在某些情况下，可以通过变量的替换将无穷区间上的积分变成一个有限区间上的积分，这样就可以采用适当的方法进行求积.例如,对积分

$$\int_0^\infty f(x)dx$$

令 $t=e^{-x}$（即 $x=-\ln t$），则

4.11.2 $$\int_0^\infty f(x)dx=\int_0^1 \frac{1}{t}f(-\ln t)dt$$

上式右边即是有限区间上的积分.

4.11.2 截去无穷区间

适当选取 $R>a$，使

$$\left|\int_R^\infty f(x)dx\right|<e$$

其中 e 为允许的误差。那么，在无穷区间上的积分可以用

$\int_a^R f(x)dx$ 来近似。

4.11.3 例 计算

$$\int_0^\infty e^{-x^2}dx$$

解 当 $x\geqslant R$ 时有 $x^2\geqslant Rx$，所以有估计式

$$\int_R^\infty e^{-x^2}dx\leqslant\int_R^\infty e^{-Rx}dx=\frac{1}{R}e^{-R^2}$$

对于 $R=4$，则 $\frac{1}{R}e^{-R^2}\approx10^{-8}$。因此对于允许误差为 10^{-7} 来说，只要计算 $\int_0^4 e^{-x^2}dx$ 就可以了。

4.12

4.12 重积分的数值计算

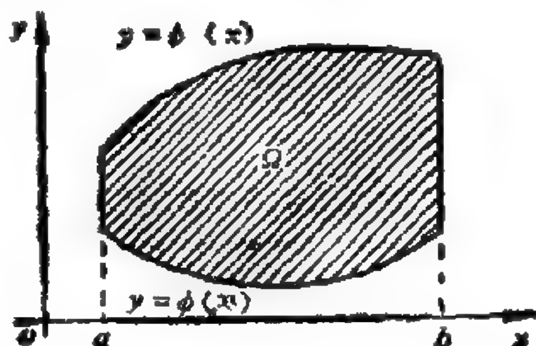
4.12.1 基本概念

设函数 f 在某一有界区域 Ω 内有定义并且是连续的，要计算积分

4.12.1 $I(f)=\iint\limits_{\Omega}f(x,y)dxdy$

假定积分区域 Ω 是由二条连续单值的曲线 $y=\varphi(x)$，$y=\psi(x)$ （$\varphi(x)\leqslant\psi(x)$），$a\leqslant x\leqslant b$ 和二条垂直线 $x=a$，$x=b$ 围成的（见图 4.12.2）

4.12.2 图



可以把积分（4.12.1）表成

$$\iint\limits_{\Omega} f(x,y)dxdy = \int_a^b dx \int_{\varphi(x)}^{\psi(x)} f(x,y)dy$$

假定

$$F(x) = \int_{\varphi(x)}^{\psi(x)} f(x,y)dy$$

那么有

$$\iint\limits_{\Omega} f(x,y)dxdy = \int_a^b F(x)dy$$

上式右边的定积分可以采用数值方法求积得到

$$\iint\limits_{\Omega} f(x,y)dxdy = \sum_{k=1}^{n} C_k F(x_k) \tag{4.12.3}$$

其中 $x_k \in [a, b]$ $(k=1, 2, \cdots, n)$，C_k 为求积系数。

对于

$$F(x_k) = \int_{\varphi(x_k)}^{\psi(x_k)} f(x_k, y)dy$$

也可以用求积公式

$$F(x_k) = \sum_{l=1}^{m_k} B_{kl} f(x_k, y_l)$$

来求得，其中 B_{kl} 是适当的常数。

由（4.12.3）可以得到

$$\iint\limits_{\Omega} f(x,y)dxdy = \sum_{k=1}^{n} \sum_{l=1}^{m_k} C_k B_{kl} f(x_k, y_l) \tag{4.12.4}$$

此公式称为乘积型求积公式。

4.12.2 梯形公式及其复化公式

设积分区域是矩形

$R=\{(x,y)|\ a\leqslant x\leqslant A,\ b\leqslant y\leqslant B\}$

它的每一边平行于坐标轴．令

$x_0=a,\ x_1=A,\ y_0=b,\ y_1=B$

于是得到四个点$(x_k,\ y_l)$ $(k,l=0,1)$。如果f在R内连续，则有

4.12.5 $$\iint\limits_R f(x,y)dxdy=\int_a^A dx\int_b^B f(x,y)dy$$

利用梯形公式计算内部积分

$$\iint\limits_R f(x,y)dxdy=\frac{B-b}{2}\int_a^A [f(x,y_0)+f(x,y_1)]dx$$

对上式右边再次应用梯形公式，可得

4.12.6 $$\iint\limits_R f(x,y)dxdy=\frac{1}{4}(B-b)(A-a)[f(x_0,y_0)+f(x_1,y_0)+f(x_0,y_1)+f(x_1,y_1)]$$

此公式称作梯形公式。

为了提高精度，可以采用复化求积公式，即把求积区域R划分为一组矩形，而在每个矩形上应用梯形求积公式。

设把矩形R的边分别分为n等分和m等分，这样便把R分为边长为h和k的mn个小矩形。在每个小矩形上应用矩形公式得

$$\iint\limits_R f(x,y)dxdy\approx\frac{hk}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}[f(x_i,y_j)$$

$$+f(x_i,y_{j+1})+f(x_{i+1},y_j)+f(x_{i+1},y_{j+1})]$$

其中 $x_i=ih\ (i=0,1,\cdots,n),y_j=jk\quad(j=0,1,\cdots,m)$

上式可以改写为

$$4.12.7\quad \iint\limits_R f(x,y)dx\approx\frac{hk}{4}\sum_{i=0}^{n}\ \sum_{j=0}^{m}\lambda_{ij}f(x_i,y_j)$$

其中 λ_{ij} 是下面矩阵 A 的相应的元素，

$$A=\begin{pmatrix}1&2&2&\cdots&2&2&1\\2&4&4&\cdots&4&4&2\\2&4&4&\cdots&4&4&2\\\cdots&\cdots&\cdots&\cdots&\cdots&\cdots\\2&4&4&\cdots&4&4&2\\2&4&4&\cdots&4&4&2\\1&2&2&\cdots&2&2&1\end{pmatrix}$$

公式（4.12.7）称为**复化梯形公式**

4.12.3 Simpson 公式及其复化公式

取积分区域为

$R=\{(x,y)|\ \ a\leqslant x\leqslant A,\ b\leqslant y\leqslant B\}$

分别用点

$x_0=a,\ x_1=a+h,\ x_2=a+2h=A$

和

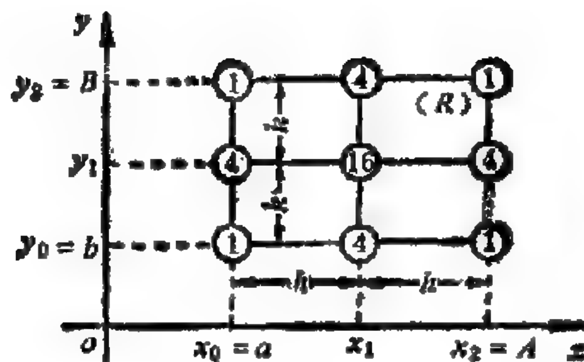
$y_0=b,\ y_1=b+h,\ y_2=b+2k=B$

于分区间$[a,A]$和$[b,B]$，其中

$$h=\frac{1}{2}(A-a),\quad k=\frac{1}{2}(B-b)$$

这样得到九个点$(x_i,\ y_j)$（$i,j=0,\ 1,\ 2$），点的分布见图（4.12.8）。

4.12.8 图



利用（4.12.5），并对内部积分用 Simpson 求积公式，

$$\iint\limits_R f(x,y)dxdy$$

$$=\frac{k}{3}\left[\int_a^A f(x,y_0)dx+4\int_a^A f(x,y_1)dx+\int_a^A f(x,y_2)dx\right]$$

再对上式右边的每个积分应用 Simpson 求积公式，有

4.12.9 $$\iint\limits_R f(x,y)dxdy\approx\frac{kh}{9}\{[f(x_0,y_0)+f(x_0,y_2)+f(x_2,y_0)+f(x_2,y_2)]+4[f(x_1,y_0)+f(x_0,y_1)+f(x_2,y_1)+f(x_1,y_2)]+16f(x_1,y_1)\}$$

此公式称作 Simpson 公式．如果令 σ_0 为被积函数 f 在矩形 R 的角点上的值之和，σ_1 为 f 在矩形 R 的每边中点上的值之和，σ_2 是 f 在矩形 R 的中心上的值，那么公式（4.12.9）可以表示为

$$\iint\limits_R f(x,y)dxdy=\frac{kh}{9}(\sigma_0+4\sigma_1+16\sigma_2)$$

上式右边的 $\sigma_i(i=0,1,2)$系数见图（4.12.8）．

为提高求积精度，一般采用复化公式．设把矩形 R 的

每边分别分成 n 等分和 m 等分，这就得到了 nm 个小矩形。再把每个小矩形等分为四部分，这样就把 R 剖分成更小的矩形，并把这些矩形的顶点用作求积公式中的节点。

令

$$h=\frac{A-a}{2n}, \qquad k=\frac{B-a}{2m}$$

那么节点的坐标为

$$x_i=x_0+ih \quad (x_0=a, i=0,1,\cdots,2n)$$

$$y_j=y_0+jh \quad (y_0=b, j=0,1,\cdots,2m)$$

在第一次分 R 的 nm 个矩形上应用公式（4.12.9）并记 $f_{ij}=f(x_i, y_j)$后，有

$$\iint_R f(x,y)dxdy \approx \frac{hk}{9}\sum_{i=0}^{n-1}\sum_{j=0}^{m-1}[(f_{2i,2j}+f_{2i+2,2j}+f_{2i+2,2j+2}+f_{2i,2j+2})+4(f_{2i+1,2j}+f_{2i+2,2j+1}+f_{2i+1,2j+2}+f_{2i,2j+1})+16f_{2i+1,2j+1}]$$

改写上式可以得到

4.12.10 $$\iint_R f(x,y)dxdy \approx \frac{hk}{9}\sum_{i=0}^{2n}\sum_{j=0}^{2m}\lambda_{ij}f_{ij}$$

其中系数 λ_{ij} 是矩阵 A 的相应的元素。A 定义为

$$A=\begin{pmatrix} 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & 1 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 16 & 8 & 16 & 4 \\ 2 & 8 & 4 & 8 & 4 & \cdots & 8 & 4 & 8 & 2 \\ \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ 2 & 8 & 4 & 8 & 4 & \cdots & 8 & 4 & 8 & 2 \\ 4 & 16 & 8 & 16 & 8 & \cdots & 16 & 8 & 16 & 8 \\ 1 & 4 & 2 & 4 & 2 & \cdots & 4 & 2 & 4 & 1 \end{pmatrix}$$

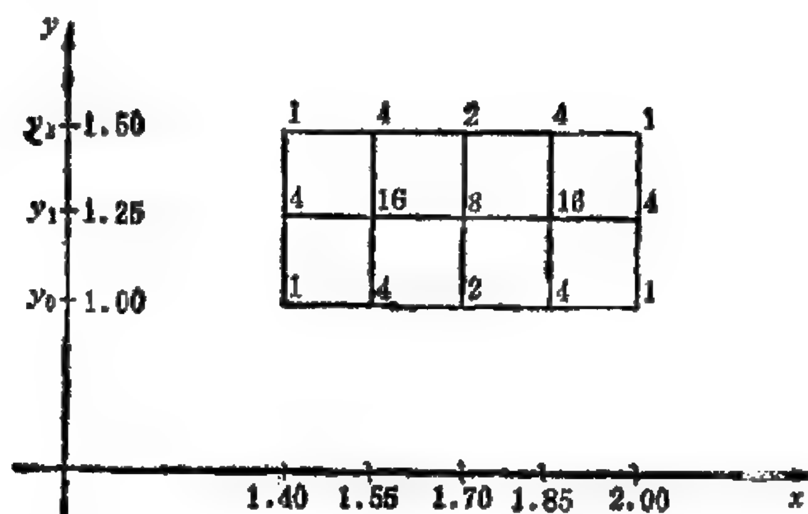
4.12.11 **例** 用复化 Simpson 公式计算积分

$$\iint_R \ln(x+2y)dxdy \quad (\text{精确值为}0.4295545265)$$

其中 $R=\{(x,y)|1.4\leqslant x\leqslant 2.0,\ 1.0\leqslant y\leqslant 1.5\}$.

解 先剖分R，取$h=0.15$，$k=0.25$，则有$n=2$，$m=1$。节点(x_i,y_j) $i=0,1,2,3,4$，$j=0,1,2$ 分布见图(4.12.12)。

4.12.12 图



利用求积公式(4.12.10)，$f_{ij}=\ln(x_i+2y_j)$，系数 λ_{ij} 在图(4.12.11)上标出，则有

$$\iint_R \ln(x+2y)dydx=\int_{1.4}^{2.0}dx\int_{1.0}^{1.5}\ln(x+2y)dy$$

$$\approx\frac{(0.15)(0.25)}{9}\sum_{i=0}^{4}\sum_{j=0}^{2}\lambda_{ij}\ln(x_i+2y_j)=0.4295524387$$

此计算值与精确值相比精确到 2.1×10^{-6}。

4.12.4 Gauss型求积公式

在 Gauss-Legendre 求积公式(见4.5)中，

$$\int_{-1}^{1} g(t)dt \approx \sum_{i=1}^{n} A_i g(t_i)$$

对$2n-1$次的代数多项式是精确成立的，上式中的节点 t_i 及系数 A_i 见(4.16)附表(4.16.1)，定积分的 Gauss-Legendre 求积公式很容易推广到重积分

4.12.13 $I(f)=\int_{-1}^{1}\int_{-1}^{1} f(x,y)dxdy$

的求积。求积公式

4.12.14 $\int_{-1}^{1}\int_{-1}^{1} f(x,y)dxdy \approx \sum_{i=1}^{n} \sum_{j=1}^{n} A_i A_j f(t_i,t_j)$

对于二元函数$f(x,y)=x^{\alpha}y^{\beta}$, $-1\leqslant x, y\leqslant 1$

$0\leqslant\alpha\leqslant 2n-1, 0\leqslant\beta\leqslant 2n-1$

精确成立，其中系数 A_i 及节点 t_i仍由(4.16) 附表(4.16.1) 确定。由此可知，求积公式 (4.12.14) 对任意二元 $4n-2$ 次多项式精确成立。求积公式 (4.12.14) 也称为重积分的 Gauss 型求积公式。

4.12.15 **例** 用 Gauss 型求积公式计算例 (4.12.11) 的积分。

解 首先用线性变换

$$u=\frac{1}{2.0-1.4}(2x-1.4-2.0)$$

$$v=\frac{1}{1.5-1.0}(2y-1.0-1.5)$$

将积分区域

$R=\{(x,y)|\ 1.4\leqslant x\leqslant 2.0\ ,\ 1.0\leqslant y\leqslant 1.5\}$

变换到正方形区域

$\bar{R}=\{(x,\ y)|\ -1\leqslant u\leqslant 1,\ -1\leqslant v\leqslant 1\}$

经变换后积分为

$$\int_{1.4}^{2.0}\int_{1.0}^{1.5}\ln(x+2y)dydx$$

$$=0.75\int_{-1}^{1}\int_{-1}^{1}\ln(0.3u+0.5v+4.2)dvdu$$

对上式右边积分，使用 Gauss 求积公式 (4.12.14)，取 $n=3$，则采用节点

$u_1=v_1=0, u_0=v_0=-0.7745966692$，$u_2=v_2=0.7745966692$

相应的权 $A_1=0.8888888889$，$A_0=A_2=0.5555555556$。

于是

$$\int_{1.4}^{2.0}\int_{1.0}^{1.5}\ln(x+2y)dydx$$

$$\approx\sum_{i=0}^{2}\sum_{j=0}^{2}A_iA_j\ln(3u_i+5v_j+4.2)=0.4295545313$$

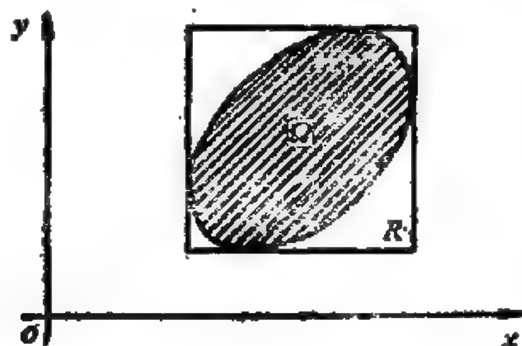
此结果与精确值相比精确到 4.8×10^{-9}。

注意到此例仅计算六次函数值而例 (4.12.11)则计算了十五次函数值，可知 Gauss 求积公式是很实用的。

4.12.5 一般积分区域

考虑积分区域为一般的曲线围成的区域 Ω，则构造一个矩形 R 使 $R\supset\Omega$，且 R 的边平行于坐标轴(见图 4.12.16)。

4.12.16 图



考虑辅助函数 f^*，其定义为

九

$$f^*(x,y)=\begin{cases} f(x,y), & \text{当}(x,y)\in\Omega \\ 0, & \text{当}(x,y)\in R-\Omega \end{cases}$$

显然有

$$\iint_{\Omega} f(x,y)dxdy=\iint_{R} f^*(x,y)dxdy$$

上式右边的积分区域为矩形，因此可以采用已介绍的各种方法来求积。

4.13

4.13 数值微分的基本方法

4.13.1 数值微分的概念

在微分学中，函数的导数是通过极限定义的，但当函数用表格给出时，就不可用定义来求其导数，只能用近似方法求数值导数。最简单的数值微分公式是用差商近似地代替微商，常见的有

$$f'(x)\approx\frac{f(x+h)-f(x)}{h}$$

$$f'(x)\approx\frac{f(x)-f(x-h)}{h}$$

$$f'(x)\approx\frac{f(x+h)-f(x-h)}{2h}$$

使用近似公式 $f'(x)\approx\frac{f(x+h)-f(x-h)}{2h}$ 的方法称为中点方法。为了使用上述近似公式的方法来近似计算微商，首先必须选取合适的步长，为此要进行误差分析。假定 f 是一个光滑的函数，利用 Taylor 展开有

$$f(x\pm h)=f(x)\pm hf'(x)+\frac{h^2}{2!}f''(x)$$
$$\pm\frac{h^3}{3!}f'''(x)+\frac{h^4}{4!}f^{(4)}(x)\pm\frac{h^5}{5!}f^{(5)}(x)+\cdots$$

如果用中点公式进行分析，则有

$$\frac{f(x+h)-f(x-h)}{2h}=f'(x)+\frac{h^2}{3!}f'''(x)+\frac{h^4}{5!}f^{(5)}(x)+\cdots$$

故步长越小，结果越准确。但是，事实并非如此，因为当h很小时，由于 $f(x+a)$ 与 $f(x-a)$很接近，直接相减会造成有效数字的严重损失，所以必须适当选取 h。

4.13.1 例 用中点公式计算$f(x)=\sqrt{x}$在 $x=2$ 处的一阶导数（导数精确值 $f'(2)=0.353553$）。

解 计算采用公式

$$\frac{\sqrt{2+h}-\sqrt{2-h}}{2h}\approx f'(2)$$

取 4 位数字计算，结果见表（4.13.2）

4.13.2 表

h	$\frac{1}{2h}(\sqrt{2+h}-\sqrt{2-h})$	h	$\frac{1}{2h}(\sqrt{2+h}-\sqrt{2-h})$	h	$\frac{1}{2h}(\sqrt{2+h}-\sqrt{2-h})$
1	0.3660	0.05	0.3530	0.001	0.3500
0.5	0.3564	0.01	0.3500	0.0005	0.3000
0.1	0.3535	0.005	0.3500	0.0001	0.3000

可以看出，$h=0.1$ 的逼近效果最好，如果 h 再缩小，则逼近的效果反而差。

4.13.2 用插值多项式求数值导数

设 f 是定义在$[a,b]$上的函数，并在$[a,b]$上给定 $n+1$ 个节点 $x_0, x_1, \cdots, x_n$，则可以求得 f 在这些节点上的插值多项式 P_n。若取 Lagrange 形式，则

$$P_n(x)=\sum_{i=0}^{n} f(x_i)l_i(x), \qquad x\in[a,b]$$

其中

$$l_i(x)=\prod_{\substack{j=0\\j\neq i}}^{n}\left(\frac{x-x_j}{x_i-x_j}\right)\ (i=0,1,\cdots,n)$$

利用插值多项式 P_n，可以把 f 表示为

$f=P_n+R_n$

其中 R_n 为插值多项式 P_n 的余项。如果 f 有 $n+1$ 阶的连续导数，则 Lagrange 形式的插值多项式的余项为

$$R_n(x)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\prod_{j=0}^{n}(x-x_j),\ \xi\in(a,b)$$

由此可以得到

$$f'(x)=P_n'(x)+\frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{d}{dx}\left(\prod_{j=0}^{n}(x-x_j)\right)$$

$$+\frac{1}{(n+1)!}\prod_{j=0}^{n}(x-x_j)\frac{d}{dx}f^{(n+1)}(\xi)$$

式中 ξ 是 x 的未知函数，因此无法对上式右边的第三项进行估计，但取 x 为插值点 x_k 时，$f'(x)\approx P_n'(x)$，其误差估计为

$$R_n'(x_k)=\frac{f^{(n+1)}(\xi)}{(n+1)!}\frac{d}{dx}\left(\prod_{\substack{j=0\\j\neq k}}^{n}(x_k-x_j)\right)$$

在等距节点的情况下，设间距为 h，令 $x=x_0+th$，利用 Newton 前插公式有

$$P_n(x)=y_0+t\Delta y_0+\frac{t(t-1)}{2!}\Delta^2 y_0+\cdots$$

$$+\frac{t(t-1)\cdots(t-n+1)}{n!}\Delta^n y_0$$

其中 $y_0=f(x_0)$。于是得到

4.13.3 $f'(x_0)\approx P_n'(x_0)$

$$=\frac{1}{h}\left[\Delta y_0-\frac{1}{2}\Delta^2 y_0+\frac{1}{3}\Delta^3 y_0+\cdots+\frac{(-1)^{n-1}}{n}\Delta^n y_0\right]$$

此式适用于表头。同样，适用于表末($t\leqslant 0$)的公式是

4.13.4 $f'(x_0)$

$$\approx\frac{1}{h}\left(\Delta y_{-1}+\frac{1}{2}\Delta^2 y_{-2}+\frac{1}{3}\Delta^3 y_{-3}+\cdots+\frac{1}{n}\Delta^n y_{-n}\right)$$

对于表中间，则采用 Stirling 插值多项式的微商，

4.13.5 $f'(x_0)\approx\frac{1}{h}\left(\frac{\Delta y_{-1}+\Delta y_0}{2}-\frac{1}{6}\frac{\Delta^3 y_{-2}+\Delta^3 y_{-1}}{2}\right.$

$$\left.+\frac{1}{30}\frac{\Delta^5 y_{-3}+\Delta^5 y_{-2}}{2}+\cdots\right)$$

在实际计算中，使用公式（4.13.5）更方便，并且比公式（4.13.3）和公式（4.13.4）有更高的精度。

如果在公式(4.13.3)和公式（4.13.4)中仅取一项，则有

4.13.6 $$\begin{cases}f'(x_0)\approx\frac{1}{h}(y_1-y_0)\\ f'(x_0)\approx\frac{1}{h}(y_0-y_{-1})\end{cases}$$

此式称为**二点公式**。

如果在公式（4.13.3）和公式（4.13.4）中取前面二项，而在公式（4.13.5）中取一项，则得到**三点公式**

4.13.7
$$\begin{cases} f'(x_0) \approx \dfrac{1}{2h}(-3y_0+4y_1-y_2) \\ f'(x_0) \approx \dfrac{1}{2h}(y_{-2}-4y_{-1}+3y_0) \\ f'(x_0) \approx \dfrac{1}{2h}(y_1-y_{-1}) \end{cases}$$

此外，常用的还有五点公式

4.13.8
$$\begin{cases} f'(x_0) \approx \dfrac{1}{12h}(-25y_0+48y_1-36y_2+16y_3-3y_4) \\ f'(x_0) \approx \dfrac{1}{12h}(-3y_{-1}-10y_0+18y_1-6y_2+y_3) \\ f'(x_0) \approx \dfrac{1}{12h}(y_{-2}-8y_{-1}+8y_1-y_2) \\ f'(x_0) \approx \dfrac{1}{12h}(-y_{-3}+6y_{-2}-18y_{-1}+10y_0+3y_1) \\ f'(x_0) \approx \dfrac{1}{12h}(3y_{-4}-16y_{-3}+36y_{-2}-48y_{-1}+25y_0) \end{cases}$$

对于给定的数据表，用五点公式（4.13.8）求节点上的导数值一般可以获得好的结果.

4.13.9 例 根据$f(x)=\sqrt{x}$的数值（见表4.13.10），利用五点公式求出节点上的导数值.

4.13.10 表

x_i	$f(x_i)$	$f'(x_i)$	计算值
100	10.000000	0.050000	0.050000
101	10.049875	0.049752	0.049751
102	10.099504	0.049507	0.049507
103	10.148891	0.049266	0.049267
104	10.198039	0.049029	0.049029
105	10.246950	0.048795	0.048795

用插值多项式 P_n来代替函数 f 将产生截断误差，公式（4.13.3）和公式（4.13.4）可以用$\frac{1}{n+1}h^n|f^{(n+1)}(\xi)|$来估计，其中 ξ 分别属于区间(x_0,x_n)和(x_{-n},x_0)。通常，$f^{(n+1)}$并不知道，所以这样的截断误差估计很难应用。此外，由于舍入误差的影响，在数值微分的计算中一般不采用高阶差商，而对于仅包含低阶差商的数值微分公式，截断误差是容易估计的。例如，对于公式（4.13.7）的第三式，如果三阶差商的变化是充分光滑的，那么可以近似地用

$\frac{1}{6h}\frac{|\Delta^3 y_{-2}+\Delta^3 y_{-1}|}{2}$来估计。

利用插值多项式求数值导数时，插入多项式 P_n可收敛到 f，但 P_n'不一定收敛到 f'；此外，当 h 缩小时，截断误差减小，但舍入误差可能增加，因此计算必须注意。

4.13.3 将微分问题化为积分问题

微分是积分的逆运算，因此可借助于数值积分来计算数值微分。

设 f 是一个充分光滑的函数，其导数为 φ。由积分定义有

4.13.11 $$f(x)=f(\hat{x})+\int_{\hat{x}}^{x}\varphi(t)dt$$

其中 $\hat{x}$ 为任意指定的数。设 $x_i=x_0+ih$ $(i=0,1,\cdots,n)$ 为一组等距节点，并设 $y_k=f(x_k)$。在公式（4.13.11）中取 $\hat{x}=x_{k-1}$, $x=x_{k+1}$，于是（4.13.11）变为

4.13.12 $$f(x_{k+1})=f(x_{k-1})+\int_{x_{k-1}}^{x_{k+1}}\varphi(t)dt$$

$$(k=1,2,\cdots,n-1)$$

对上式右端的积分采用不同的求积公式就得到不同的数值微分公式。

1° 对积分采用中点公式

$$\int_{x_{k-1}}^{x_{k+1}} \varphi(t)dt = 2h\varphi(x_k) + \frac{(2h)^3}{24}\varphi''(\xi_k)$$

从而得到**中点微分公式**

4.13.13 $$f'(x_k)=\varphi(x_k)=\frac{f(x_{k+1})-f(x_{k-1})}{2h}-\frac{h^2}{6}f'''(\xi_k)$$

其中 $x_{k-1}\leqslant\xi_k\leqslant x_{k+1}$ $(k=1,2,\cdots,n-1)$。可以看出，此式与公式（4.13.7）的第三式是一样的。

2° 如果对（4.13.12）中的积分采用 Simpson 求积公式，则有

$$\int_{x_{k-1}}^{x_{k+1}} \varphi(t)dt = \frac{h}{3}[\varphi(x_{k-1})+4\varphi(x_k)+\varphi(x_{k+1})]-\frac{h^5}{90}f^{(5)}(\xi_k)$$

其中 $x_{k-1}<\xi_k<x_k$。

如果记 φ_k 为 $f'(x_k)$ 的近似值，且在上式中略去高阶项，那么从（4.13.12）可得到 Simpson **数值微分公式**

4.13.14 $$\varphi_{k-1}+4\varphi_k+\varphi_{k+1}=\frac{3(y_{k+1}-y_{k-1})}{h}\quad(k=1,2,\cdots,n-1)$$

（4.13.14）有 $n+1$ 个未知函数 $\varphi_0,\varphi_1,\cdots,\varphi_n$，但只有 $n-1$ 个方程。如果在端点有 $\varphi(x_0)=f'(x_0),\varphi(x_n)=f'(x_n)$，那么（4.13.14）可以写作

4.13.15 $$\begin{pmatrix} 4 & 1 & & & \\ 1 & 4 & 1 & & \\ & 1 & 4 & 1 & \\ \cdot & \cdot & \cdot & \cdot & \\ & & & 1 & 4 \end{pmatrix}\begin{pmatrix}\varphi_1\\ \varphi_2\\ \varphi_3\\ \vdots\\ \varphi_{n-1}\end{pmatrix}=\begin{pmatrix}3(y_2-y_0)/h & -f'(x_0)\\ 3(y_3-y_1)/h & \\ 3(y_4-y_2)/h & \\ \vdots & \\ 3(y_n-y_{n-1})/h & -f'(x_n)\end{pmatrix}$$

于是，数值微分化为解线性代数方程组的问题。显然，用追赶法容易求解。

4.13.16 **例** 给定 $f(x)=\sqrt{x}$ 在 $x=100$，101，102，103，104，105的一个表，求函数 f 在 $x=101,102,103,104$ 上的一阶导数值。

解 假定 f 在 $x=100$ 及 $x=105$ 处的一阶导数值已知，按方程组（4.13.15）解之，结果见表（4.13.17）。

4.13.17 **表**

x	$\sqrt{x}$	$(\sqrt{x})'$	导数近似值
100	10.000000	0.05000000	
101	10.049875	0.049751859	0.049751859
102	10.099504	0.049507377	0.049507376
103	10.148891	0.049266463	0.049266463
104	10.198039	0.049029033	0.049029033
105	10.246950	0.048795003	

从算例看出，采用 Simpson 数值微分公式计算的精度相当高。

如果端点的导数值并不知道，对 $\varphi(x_1)$ 和 $\varphi(x_n)$的近似值 φ_1和 φ_n则可用中点微分公式来近似，这样 Simpson 微分公式（4.13.14）可以写作

4.13.18
$$\begin{cases} \varphi_1=\dfrac{y_2-y_0}{2h} \\ \varphi_{k-1}+4\varphi_k+\varphi_{k+1}=\dfrac{3(y_{k+1}-y_{k-1})}{h} \quad (k=1,2,\cdots,n-1) \\ \varphi_{n-1}=\dfrac{y_n-y_{n-2}}{2h} \end{cases}$$

当 $n=3$ 时，有

$$
\begin{cases}
\varphi_1 = \dfrac{y_2 - y_0}{2h} \\
\varphi_0 + 4\varphi_1 + \varphi_2 = \dfrac{3(y_2 - y_0)}{h} \\
\varphi_1 + 4\varphi_2 + \varphi_3 = \dfrac{3(y_3 - y_1)}{h} \\
\varphi_2 = \dfrac{y_3 - y_1}{2h}
\end{cases}
$$

求解得

$$
\begin{cases}
f'(x_0) \approx \dfrac{1}{2h}[2(y_2 - y_0) - (y_3 - y_1)] \\
f'(x_1) \approx \dfrac{y_2 - y_0}{2h} \\
f'(x_2) \approx \dfrac{y_3 - y_1}{2h} \\
f'(x_3) \approx \dfrac{1}{2h}[2(y_3 - y_1) - (y_2 - y_0)]
\end{cases}
$$

4.13.4 用三次样条函数求数值微分

设在区间$[a,b]$上给定一个剖分

$$
a = x_0 < x_1 < \cdots < x_{n-1} < x_n = b, x_k = a + kh, \quad h = \frac{1}{n}(b-a)
$$

及相应于$\{x_k\}$的函数值$\{y_k\}$ $(k=0,1,\cdots n)$，如果给定适当的边界条件，则可以唯一确定三次样条函数S，其表达式为

$$
S(x) = \sum_{j=-1}^{n+1} c_j \Omega_3\left(\frac{x - x_j}{h}\right), \quad x \in [a,b]
$$

对上式两边求导数，则有

$$
4.13.19 \quad f'(x) \approx S'(x) = \frac{1}{h} \sum_{j=-1}^{n+1} c_j \Omega_3'\left(\frac{x - x_j}{h}\right)
$$

令 $x=x_k$，则有

$$f'(x_k)\approx S'(x_k)=\frac{1}{h}\sum_{j=-1}^{n+1}c_j\Omega_3'\left(\frac{x_k-x_j}{h}\right)$$

$$=\frac{1}{h}\sum_{j=-1}^{n+1}c_j\Omega_3'(k-j)$$

由此得

4.13.20 $f'(x_k)\approx\frac{1}{2h}(c_{k+1}-c_{k-1})\quad(k=1,2,\cdots,n-1)$

此公式说明，函数 $f(x)$ 在 x_k 处的导数值近似地由它的三次样条函数的系数 c_{k+1}, c_{k-1} 线性表示。

如果 f 有一阶连续导数，那么 S' 在 $[a,b]$ 上一致收敛到 f'，因此当 h 取得充分小时用样条求数值微分是很精确的。

4.14

4.14 二阶导数

如果 f 在区间 $[x_0-h,x_0+h]$ 上有四阶连续的导数，则利用 Taylor 级数展开有

$$f''(x_0)=\frac{1}{h^2}[f(x_0-h)-2f(x_0)+f(x_0+h)]-\frac{h^2}{12}f^{(4)}(\xi)$$

其中 $x_0-h<\xi<x_0+h$。如果略去上式右边的最后一项，可得数值微分公式

$$f''(x_0)\approx\frac{1}{h^2}[f(x_0-h)-2f(x_0)+f(x_0+h)]$$

利用插值多项式来求二阶数值微分是经常使用的。设 P_n 是 f 的插值多项式，对于任意 x，有

$$f''(x)\approx P_n''(x)$$

对于等距节点，有以下常用公式，为简便起见，记 $y_k=f(x_k)$。

4.14.1
$$\begin{cases} f''(x_0)\approx\dfrac{1}{h^2}(y_0-2y_1+y_2) \\ f''(x_0)\approx\dfrac{1}{h^2}(y_{-1}-2y_0+y_1) \\ f''(x_0)\approx\dfrac{1}{h^2}(y_{-2}-2y_{-1}+y_0) \end{cases}$$

4.14.2
$$\begin{cases} f''(x_0)\approx\dfrac{1}{6h^2}(12y_0-30y_1+24y_2-6y_3) \\ f''(x_0)\approx\dfrac{1}{6h^2}(6y_{-1}-12y_0+6y_1) \\ f''(x_0)\approx\dfrac{1}{6h^2}(-6y_{-3}+24y_{-2}-30y_{-1}+12y_0) \end{cases}$$

4.14.3
$$\begin{cases} f''(x_0)\approx\dfrac{1}{12h^2}(35y_0-104y_1+114y_2-56y_3+11y_4) \\ f''(x_0)\approx\dfrac{1}{12h^2}(11y_{-1}-20y_0+6y_1+4y_2-y_3) \\ f''(x_0)\approx\dfrac{1}{12h^2}(-y_{-2}+16y_{-1}-30y_0+16y_1-y_2) \\ f''(x_0)\approx\dfrac{1}{12h^2}(-y_{-3}+4y_{-2}+6y_{-1}-20y_0+11y_1) \\ f''(x_0)\approx\dfrac{1}{12h^2}(11y_{-4}-56y_{-3}+114y_{-2} \\ \qquad\qquad\qquad -104y_{-1}+35y_0) \end{cases}$$

二阶数值微分也可以用三次样条函数方法来计算。设在$[a,b]$上给定一个剖分

$$a=x_0<x_1<\cdots<x_n=b,\quad x_i=a+ih,\ h=\frac{b-a}{n}$$

及相应于这个剖分$\{x_i\}$的函数值$\{y_i\}$。如果再给定适当的边界条件就可以构造出唯一的三次样条函数S，其表达式为

$$S(x)=\sum_{j=-1}^{n+1}c_j\Omega_3\left(\frac{x-x_j}{h}\right)$$

已有（4.13.19）

$$f'(x)\approx S'(x)=\frac{1}{h}\sum_{j=-1}^{n+1} c_j\Omega_3'\left(\frac{x-x_j}{h}\right)$$

$$f''(x)\approx S''(x)=\frac{1}{h^2}\sum_{j=-1}^{n+1} c_j\Omega_3''\left(\frac{x-x_j}{h}\right)$$

再求一次导数，有

令 $x=x_i$，得到

$$f''(x_i)\approx S''(x_i)=\frac{1}{h^2}(c_{i+1}-2c_i+c_{i-1})$$

利用样条函数的 c_i和 y_i之间的关系式

$$\frac{1}{6}c_{i-1}+\frac{2}{3}c_i+\frac{1}{6}c_{i+1}=y_i$$

就得到

4.14.4 $f''(x_i)\approx S''(x_i)=\frac{6}{h^2}(y_i-c_i)\quad(i=1,2,\cdots,n-1)$

有些问题还可以应用下面的方法来提高精度．令

$$\widetilde{y}_i=S'(x_i)\quad(i=0,1,\cdots,n)$$
$$\widetilde{y}_0'=f''(x_0)\quad \widetilde{y}_n'=f''(x_n)$$

满足这个条件的插值函数记为$\widetilde{S'}$，

$$\widetilde{S}'(x)=\sum_{j=-1}^{n+1}\widetilde{c}_j\ \Omega_3\left(\frac{x-x_j}{h}\right)$$

对$\widetilde{S'}$求导，有

$$\widetilde{S}''(x)=\frac{1}{h}\sum_{j=-1}^{n+1}\widetilde{c}_j\ \Omega_3'\left(\frac{x-x_j}{h}\right)$$

如果以$\widetilde{S''}$来近似 f''，那么对 $x=x_i$就有

4.14.5 $f''(x_i)=\frac{1}{2h}(\widetilde{c}_{i+1}-\widetilde{c}_{i-1})$

4.15 数值微分的外推算法

在数值积分中应用外推算法产生了 Romberg 求积方法，这是非常有效的．同样，在数值微分中采用外推算法也是很有效的．

用中心差分来计算导数值有

4.15.1 $$f'(x)\approx G(h)=\frac{f\left(x+\frac{h}{2}\right)-f\left(x-\frac{h}{2}\right)}{h}$$

利用 Taylor 级数展开有

4.15.2 $$f'(x)-G(h)=a_1h^2+a_2h^4+a_3h^6+\cdots$$

其中常数a_i与h无关．利用Richardson外推算法，取$q=\frac{1}{2}$有

4.15.3 $$\begin{cases}G_1(h)=G(h)\\ G_{m+1}(h)=\dfrac{G_m\left(\frac{h}{2}\right)-\left(\frac{1}{2}\right)^{2m}G_m(h)}{1-\left(\frac{1}{2}\right)^{2m}} & (m=1,2,\cdots)\end{cases}$$

这个算法见表（4.15.4）．

4.15.4 表（①…⑩表示计算步骤）

①$G(h)$			
②$G\left(\frac{h}{2}\right)$	③$G_2(h)$		
④$G\left(\frac{h}{2^2}\right)$	⑤$G_2\left(\frac{h}{2}\right)$	⑥$G_3(h)$	
⑦$G\left(\frac{h}{2^3}\right)$	⑧$G_2\left(\frac{h}{2^2}\right)$	⑨$G_3\left(\frac{h}{2}\right)$	⑩$G_4(h)$
⋮	⋮	⋮	⋮ ⋱

利用这个算法的误差为：

$f'(x)-G_{m+1}(h)=a_{m+1}^{(m+1)}h^{2(m+1)}+\cdots$

所以，当 m 越大时越精确，但受舍入误差的影响，m不能取得很大。

4.15.5 例 利用外推算法计算 $f(x)=e^x$在 $x=1$处的导数。

$$G(h)=\frac{e^{1+\frac{h}{2}}-e^{1-\frac{h}{2}}}{h}$$

解 计算步骤及结果如下：

$G(0.8)=3.0176$

$G(0.4)=2.7914$ $G_2(0.8)=2.7160$

$G(0.2)=2.7365$ $G_2(0.4)=2.7182$ $G_3(0.8)=2.7183$

可以看出，由 $G(0.8)$，$G(0.4)$，$G(0.2)$ 经过简单运算便得到 $e(=2.7182818)$ 的很精确的近似值。

外推算法也可应用于计算二阶数值微商．仍采用中心差分公式来计算导数值

$$f''(x)\approx S(h)=\frac{f(x+h)-2f(x)+f(x-h)}{h^2}$$

利用 $f(x+h)$，$f(x-h)$在 x 处展成 Taylor 级数，经运算可得

$f''(x)-S(h)=c_1h^2+c_2h^4+c_3h^6+\cdots$

由此可以导出如同（4.15.3）的算法。

4.16

4.16 附 表

4.16.1 Gauss-Legendre 求积公式的节点和系数

求积公式为

$$\int_{-1}^{+1}f(x)dx\approx\sum_{i=1}^{n}A_if(x_i)$$

节点＝±x_i(Legendre多项式的零点)，系数＝A_i，节点数＝n.

n	x_i	A_i
2	0.57735 02691 89626	1.00000 00000 00000
3	0.00000 00000 00000	0.88888 88888 88889
	0.77495 66692 41483	0.55555 55555 55556
4	0.33998 10435 84856	0.65214 51548 62546
	0.86113 63115 94053	0.34785 48451 37454
5	0.00000 00000 00000	0.56888 88888 88889
	0.53846 93101 05683	0.47862 86704 99366
	0.90617 98459 38664	0.23692 68850 56189
6	0.23861 91860 83197	0.46791 39345 72691
	0.66120 93864 66265	0.36076 15730 48139
	0.93246 95142 03152	0.17132 44923 79170
7	0.00000 00000 00000	0.41795 91836 73496
	0.40584 51513 77397	0.38183 00505 05119
	0.74153 11855 99394	0.27970 53914 89277
	0.94910 79123 42759	0.12948 49661 68870
8	0.18343 46424 95650	0.36268 37833 78362
	0.52553 24099 16329	0.31370 66458 77887
	0.79666 64774 13627	0.22238 10344 53374
	0.96028 98564 97536	0.10122 85362 90376
9	0.00000 00000 00000	0.33023 93550 01260
	0.32425 34234 03809	0.31234 70770 40003
	0.61337 14327 00590	0.26061 06964 02935
	0.83603 11073 26636	0.18064 81606 94857
	0.96816 02395 07626	0.08127 43883 61574

续表

n	x_i	A_i
10	0.14887 43389 81631	0.29552 42247 14753
	0.43339 53941 29247	0.26926 67193 09996
	0.67940 95682 99024	0.21908 63625 15982
	0.86506 33666 88985	0.14945 13491 50581
	0.97390 65285 17172	0.06667 13443 08688
12	0.12523 34085 11469	0.24914 70458 13403
	0.36783 14989 98180	0.23349 25365 38355
	0.58731 79542 86617	0.20316 74267 23066
	0.76990 26741 94305	0.16007 83285 43346
	0.90411 72563 70475	0.10693 93259 95318
	0.98156 06342 46719	0.04717 53363 86512
16	0.09501 25098 37637 440185	0.18945 06104 55068 496285
	0.28160 35507 79258 913230	0.18260 34150 44923 588867
	0.45801 67776 57227 386342	0.16915 65193 95002 538189
	0.61787 62444 02643 748447	0.14959 59888 16576 732081
	0.75540 44083 55003 033895	0.12462 89712 55533 872052
	0.86563 12023 87831 743880	0.09515 85116 82492 784810
	0.94457 50230 73232 576078	0.06225 35239 38647 892863
	0.98940 09349 91649 932596	0.02715 24594 11754 094852
20	0.07652 65211 33497 333755	0.15275 33871 30725 850698
	0.22778 58511 41645 078080	0.14917 29864 72603 746788
	0.37370 60887 15419 560673	0.14209 61093 18382 051329
	0.51086 70019 50827 098004	0.13168 86384 49176 626898
	0.63605 36807 26515 025453	0.11819 45319 61518 417312
	0.74633 19064 60150 792614	0.10193 01198 17240 435037
	0.83911 69718 22218 823395	0.08327 67415 76704 748725
	0.91223 44282 51325 905868	0.06267 20483 34109 063570
	0.96397 19272 77913 791268	0.04060 14298 00386 941331
	0.99312 85991 85094 924786	0.01761 40071 39152 118312

4.16.2 Gauss-Laguerre求积公式的节点和系数

求积公式为

$$\int_0^{\infty} e^{-x} f(x) \approx \sum_{i=1}^{n} A_i f(x_i), \int_0^{\infty} g(x)dx \approx \sum_{i=1}^{n} A_i e^{x_i} g(x_i)$$

节点$=x_i$(Laguerre 多项式的零点)，系数$=A_i$，节点数$=n$。

n	x_i	A_i	$A_i e^{x_i}$
2	0.58578 64376 27	0.853553 390593	1.53332 603312
	3.41421 35623 73	0.146446 609407	4.45095 733505
3	0.41577 45567 83	0.711093 009929	1.07769 285927
	2.29428 03602 79	0.278517 733569	2.76214 296190
	6.28994 50829 37	0.103892 565016$\times 10^{-1}$	5.60109 462543
4	0.32254 76896 19	0.603154 104342	0.83273 91238 38
	1.74576 11011 58	0.357418 692438	2.04810 243845
	4.53662 02969 21	0.388879 085150$\times 10^{-1}$	3.63114 630582
	9.39507 09123 01	0.539294 705561$\times 10^{-3}$	6.48714 508441
5	0.26356 03197 18	0.521755 610583	0.67909 40422 08
	1.41340 30591 07	0.398666 811083	1.63848 787360
	3.59642 57710 41	0.759424 496817$\times 10^{-1}$	2.76944 324237
	7.08581 00058 59	0.361175 867992$\times 10^{-2}$	4.31565 690092
	12.64080 08442 76	0.233699 723858$\times 10^{-4}$	7.21918 635435
6	0.22284 66041 79	0.458964 673950	0.57353 55074 23
	1.18893 21016 73	0.417000 830772	1.36925 259071
	2.99273 63260 59	0.113373 382074	2.26068 459338
	5.77514 35691 05	0.103991 974531$\times 10^{-1}$	3.35052 458236
	9.83746 74183 83	0.261017 202815$\times 10^{-3}$	4.88682 680021
	15.98287 39806 02	0.898547 906430$\times 10^{-6}$	7.84901 594560

续表

n	x_i	A_i		$A_i e^{x_i}$	
7	0.19304 36765 60	0.409318	951701	0.49647	75975 40
	1.02666 48953 39	0.421831	277862	1.17764	306086
	2.56787 67449 51	0.147126	348658	1.91824	978166
	4.90035 30845 26	0.206335	144687×10^{-1}	2.77184	863623
	8.18215 34445 63	0.107401	014328×10^{-2}	3.84124	912249
	12.73418 02917 98	0.158654	643486×10^{-4}	5.38067	820792
	19.39572 78622 63	0.317031	547900×10^{-7}	8.40543	248683
8	0.17027 96323 05	0.369188	589342	0.43772	34104 93
	0.90370 17767 99	0.418786	780814	1.03386	934767
	2.25108 66298 66	0.175794	986637	1.66970	976566
	4.26670 01702 88	0.333434	922612×10^{-1}	2.37692	470176
	7.04590 54023 93	0.279453	623523×10^{-2}	3.20854	091335
	10.75851 60101 81	0.907650	877336×10^{-4}	4.26857	551083
	15.74067 86412 78	0.848574	671627×10^{-6}	5.81808	336867
	22.86313 17368 89	0.104800	117487×10^{-8}	8.90622	621529
9	0.15232 22277 32	0.336126	421798	0.39143	11243 16
	0.80722 00227 42	0.411213	980424	0.92180	50285 29
	2.00513 51556 19	0.199287	525371	1.48012	790994
	3.78347 39733 31	0.474605	627657×10^{-1}	2.08677	080755
	6.20495 67778 77	0.559962	661079×10^{-2}	2.77292	138971
	9.37298 52516 88	0.305249	767093×10^{-3}	3.59162	606809
	13.46623 69110 92	0.659212	302608×10^{-5}	4.64876	600214
	18.83359 77889 92	0.411076	933035×10^{-7}	6.21227	541975
	26.37407 18909 27	0.329087	403035×10^{-10}	9.36321	823771

10	0.13779 34705 40	0.308441 115766	0.35400 97386 07
	0.72945 45495 03	0.401119 929115	0.83190 23010 44
	1.80834 29017 40	0.218068 287612	1.33028 856175
	3.40143 36978 55	0.620874 560987×10^{-1}	1.86306 390311
	5.55249 61400 64	0.950151 697518×10^{-2}	2.45025 555808
	8.33015 27467 64	0.753008 388588×10^{-3}	3.12276 415514
	11.84378 58379 00	0.282592 334960×10^{-4}	3.93415 269556
	16.27925 78313 78	0.424931 398496×10^{-6}	4.99241 487219
	21.99658 58119 81	0.183956 482398×10^{-8}	6.57220 248513
	29.92069 70122 74	0.991182 721961×10^{-11}	9.78469 584037

4.16.3 Gauss-Hermite 求积公式的节点和系数

求积公式为

$$\int_{-\infty}^{\infty} e^{-x^2} f(x)dx \approx \sum_{i=1}^{n} A_i f(x_i),\ \int_{-\infty}^{\infty} g(x)dx = \sum_{i=1}^{n} A_i e^{x_i^2} g(x_i)$$

节点$=\pm x_i$(Hermite 多 项 式 的 零 点)，系 数$=A_i$，节 点数$=n$。

n	x_i	A_i	$A_i e^{x_i^2}$
2	0.70710 67811 86548	0.886226 92545 28	1.46114 11826 611
3	0.00000 00000 00000	1.18163 59006 04	1.18163 59006 037
	1.22474 48713 91589	0.295409 97515 09	1.32393 11752 136
4	0.52464 76232 75290	0.804914 09000 55	1.05996 44828 950
	1.65068 01238 85785	0.813128 35447 25×10^{-1}	1.24022 58176 958
5	0.00000 00000 00000	0.945308 72048 29	0.94530 87204 829
	0.95857 24646 13819	0.393619 32315 22	0.98658 09967 514
	2.02018 28704 56086	0.199532 42059 05×10^{-1}	1.18148 86255 360

续表

n	x_i	A_i	$A_i e^{x_i^2}$
6	0.43607 74119 27617	0.724629 59522 44	0.87640 13344 362
	1.33584 90740 13697	0.157067 32032 29	0.93558 05576 312
	2.35060 49736 74492	0.453000 99055 09 $\times 10^{-2}$	1.13690 83326 745
7	0.00000 00000 00000	0.810264 61755 68	0.81026 46175 568
	0.81628 78828 58965	0.425607 25231 01	0.82868 73032 836
	1.67355 16287 67471	0.545155 82819 13 $\times 10^{-1}$	0.89718 46002 252
	2.65196 13568 35233	0.971781 24509 95 $\times 10^{-3}$	1.10133 07296 103
8	0.38118 69902 07322	0.661147 01255 82	0.76454 41286 517
	1.15719 37124 46780	0.207802 32581 49	0.79289 00483 864
	1.98165 67566 95843	0.170779 83007 41 $\times 10^{-1}$	0.86675 26065 634
	2.93063 74202 57244	0.199604 07221 14 $\times 10^{-3}$	1.07193 01442 480
9	0.00000 00000 00000	0.720235 21560 61	0.72023 52156 061
	0.72355 10187 52838	0.432651 55900 26	0.73030 24527 451
	1.46855 32892 16668	0.884745 27394 38 $\times 10^{-1}$	0.76460 81250 946
	2.26658 05845 31843	0.494362 42755 37 $\times 10^{-2}$	0.84175 27014 787
	3.19099 32017 81528	0.396069 77263 26 $\times 10^{-4}$	1.04700 35809 767
10	0.34290 13272 23705	0.610862 63373 53	0.68708 18539 513
	1.03661 08297 89514	0.240138 61108 23	0.70329 63231 049
	1.75668 36492 99882	0.338743 94455 48 $\times 10^{-1}$	0.74144 19319 436
	2.53273 16742 32790	0.134364 57467 81 $\times 10^{-2}$	0.82066 61264 048
	3.43615 91188 37738	0.764043 28552 33 $\times 10^{-5}$	1.02545 16913 657
12	0.31424 03762 54359	0.570135 23626 25	0.62930 78743 695
	094778 83912 40164	0.260492 31026 42	0.63962 12320 203
	1.59768 26351 52605	0.516079 85615 88 $\times 10^{-1}$	0.66268 27732 669
	2.27950 70805 01060	0.390539 05846 29 $\times 10^{-2}$	0.70522 03661 122
	3.02063 70251 20890	0.857368 70435 88 $\times 10^{-4}$	0.78664 39394 633
	3.88972 48978 69782	0.265855 16843 56 $\times 10^{-6}$	0.98969 90470 923

5

第5章　方程求根

5.1

5.1 引　言

在科学技术的数学问题中，经常遇到求解函数方程

5.1.1　$f(x)=0$

这里$f(\cdot)$是单变量x的函数，它可以是代数多项式，即

5.1.2　$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n\quad(a_0\neq 0)$

也可以是超越函数.满足$f(x^*)=0$的值x^*就是方程(5.1.1)的解，x^*称作方程的根，又称作函数$f(\cdot)$的零点.如果$f(x)$可分解为

5.1.3　$f(x)=(x-a)^m g(x)$

且$g(a)\neq 0$，则称a为$f(x)=0$的m重根．$m=1$称为单根，$m>1$称为重根.

方程求根应解决三个问题：

1°　判断根是否存在(方程有没有根？如果有根，有几个根？)．如果$f(\cdot)$是n次代数多项式，则由代数基本定理可知，在复数域内方程有n个根.对一般函数方程，若$f(\cdot)$在区间$[a,b]$上连续，且$f(a)f(b)<0$，则方程(5.1.1)在$[a,b]$内至少有一实根，$[a,b]$为一个有根区间.

2°　根的隔离.求出每个根所在的子区间(或子区域)，若每个子区间(或子区域)只有一个根，这叫根的隔离.做好根的隔离就得到方程各根的近似值.

3°　根的精确化.已知一个根的近似值后再把根精确化，直到足够精确为止.

5.2

5.2 实根的隔离与二分法

假定$f(\cdot)$在$[a, b]$上连续，且满足条件

5.2.1 $f(a)f(b)<0$

则$[a, b]$是有根区间，于是可从$x_0=a$出发取步长$h=\frac{b-a}{n}$(n为正整数)，令 $x_k=a+kh$ ($k=0,1, \cdots,n$)，从左至右检查$f(x_k)$的符号，如发现节点x_k与端点a的函数值异号，则得到一个缩小的有根区间$[x_{k-1}, x_k]$，其宽度为h. 再检查下去，只要发现相邻两点函数值异号则可得一个缩小的有根区间.

5.2.2 例 给定方程

$f(x)=x^3-x-1=0$

由于$f(0)<0$，$f(2)>0$，故$[0, 2]$是有根区间. 若取$h=0.5$，从左向右检查$f(x_k)$的符号（见下表），可发现$[1,1.5]$是一个缩小的有根区间.

x_k	0	0.5	1.0	1.5	2.0
$f(x_k)$	−	−	−	+	+

用这种逐步搜索的方法进行实根隔离的关键是选取步长h，如h选得太大，则有的根可能被遗漏；如h选得太小，则可得到较精确的有根区间，但工作量较大. 要选择适当h，使之既能把根隔离开来，工作量又不太大，可采用二分法(Bisection method)，它可看作逐步搜索法的改进.

设$f(\cdot)$在$[a, b]$连续，假定$f(a)<0$，$f(b)>0$，取中点$x_0=\frac{a+b}{2}$，检查$f(x_0)$符号. 若$f(x_0)=0$，则x_0就是一个根；若

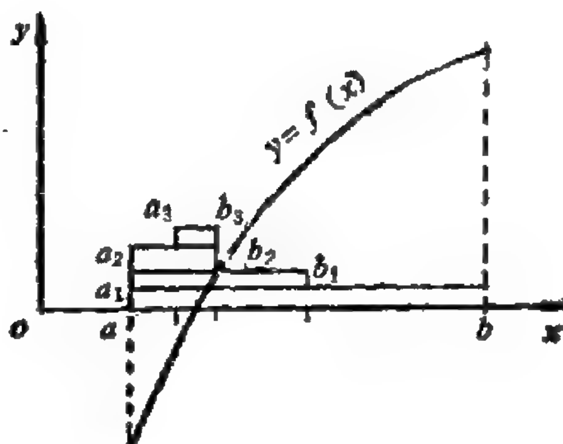
$f(x_0)>0$，记a为a_1，x_0为b_1，则得有根区间$[a_1, b_1]$；若$f(x_0)<0$，记x_0为a_1，b为b_1，则得有根区间$[a_1, b_1]$。后两种情况都得到有根区间$[a_1, b_1]$，它的长度为原区间的一半。

对$[a_1, b_1]$，令$x_1=\frac{a_1+b_1}{2}$，再施以同样方法，可得新的有根区间$[a_2, b_2]$，它的长度为$[a_1, b_1]$的一半，如此反复进行下去可得到一系列有根区间

$[a, b]\supset[a_1, b_1]\supset\cdots\supset[a_n, b_n]\supset\cdots$

其中每一个区间都是前一区间的一半，见图(5.2.3)。

5.2.3 图



因此，$[a_n, b_n]$的长度为

$$b_n-a_n=\frac{b-a}{2^n}$$

当$n\to\infty$时趋于零，且$\lim\limits_{n\to\infty}x_n=\lim\limits_{n\to\infty}\frac{a_n+b_n}{2}=x^*$，这就是方程的根。而$x_n=\frac{a_n+b_n}{2}$即为方程的近似根，且有误差估计

5.2.4 $$|x_n-x^*|\leqslant\frac{b-a}{2^{n+1}}$$

5.2.5 **例** 用二分法求

$f(x)=x^3-x-1=0$

在区间〔1，1.5〕的一个实根，准确到小数点后第2位数.

解 这里$a=1$，$b=1.5$，根据上述步骤取区间中点$x_0=1.25$，检查$f(x_0)$符号，决定新区间.如此反复得到一系列区间如下：

	有根区间
$f(1)<0$	
$f(1.5)>0$	〔1,1.5〕
$f(1.25)<0$	〔1.25,1.5〕
$f(1.375)>0$	〔1.25,1.375〕
$f(1.3125)<0$	〔1.3125,1.375〕
$f(1.34375)>0$	〔1.3125,1.34375〕
$f(1.3281)>0$	〔1.3125,1.3281〕
$f(1.3203)<0$	〔1.3203,1.3281〕

取$x_6=1.3242$，误差限$|x_6-x^*|<\frac{0.5}{2^7}<0.005$，故$x_6$即为所求近似根，实际上根$x^*=1.324717\cdots\cdots$.

上述二分法优点是计算简单，收敛性有保证，缺点是收敛不够快，特别是精度要求高时工作量大，而且，不能求复根及双重根.

5.3

5.3 迭代法

5.3.1 迭代法及其收敛性

已知方程$f(x)=0$的一个近似根x_0后，通常可用一种迭代法对这个近似根逐步精确化，一直达到精度要求为止.为了构造迭代公式，通常可把方程（5.1.1）写成等价形式

5.3.1 $x=g(x)$

并得到迭代序列

5.3.2 $x_{k+1}=g(x_k)$ $(k=0,1,\cdots)$

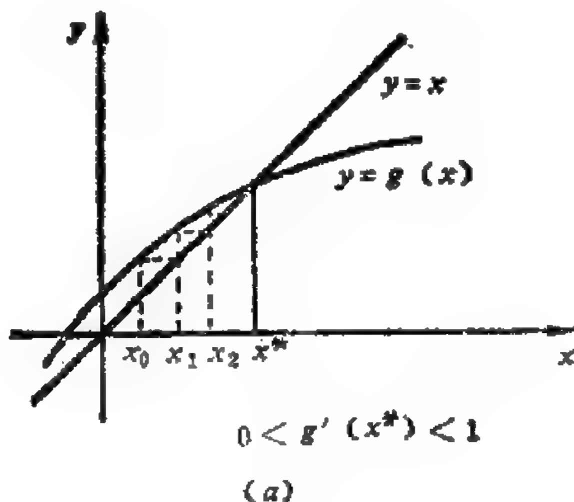
如果序列$\{x_k\}$有极限

$$x^* = \lim_{k\to\infty} x_k$$

则称迭代过程（5.3.2）收敛．这时的x^*就是方程(5.3.1)的根，由于构造的迭代函数$g(\cdot)$不一定保证序列（5.3.2）都收敛，因此，迭代前必须研究迭代法的收敛性．

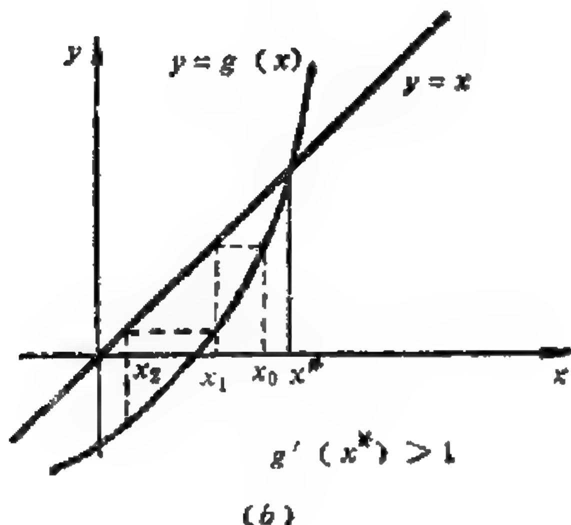
方程(5.3.1)的求根问题，从几何图象考察就是在xy平面上确定曲线$y=x$与$y=g(x)$的交点P^*．用迭代法(5.3.2)求根就是从$y=x_0$与$y=g(x)$交点出发逐次求点P^*的横坐标x^*，图（5.3.3)(a）表示迭代序列(5.3.2)收敛，(b）表示迭代不收敛．

5.3.3 图



$0 < g'(x^*) < 1$

(a)



$g'(x^*) > 1$

(b)

5.3.4 例 用迭代法求方程

$$f(x) = x^3 - x - 1 = 0$$

在$x_0 = 1.5$附近的根x^*．

解 将方程改写成

$x=\sqrt[3]{1+x}$

并建立迭代程序

5.3.5 $x_{k+1}=\sqrt[3]{1+x_k}\quad(k=0,1,\cdots)$

只要按（5.3.5）逐步计算$g(x_k)=\sqrt[3]{1+x_k}$，便可得到迭代结果：

$x_0=1.5$, $x_1=1.35721$, $x_2=1.33086$

$x_3=1.32588$, $x_4=1.32494$, $x_5=1.32476$

$x_6=1.32473$, $x_7=1.32472$, $x_8=1.32472$

这说明迭代序列（5.3.5）收敛，且$x_8=1.32472$为方程的近似根.但如果将方程改写为$x=x^3-1$，并建立迭代公式

5.3.6 $x_{k+1}=x_k^3-1$

则$x_0=1.5$，$x_1=2.375$，$x_2=12.39$，$\cdots$. x_k的值越算越大，说明迭代序列（5.3.6）不收敛，因此这个迭代序列不能用。

若（5.3.1）在$[a, b]$上有根x^*，则由迭代（5.3.2）得

5.3.7 $x_{k+1}-x^*=g(x_k)-g(x^*)=g'(\xi)(x_k-x^*)$

ξ在x_k与x^*之间，当然$\xi\in[a, b]$。因此，当$|g'(x)|\leqslant L<1$时，由（5.3.7）可得

$|x_{k+1}-x^*|\leqslant L|x_k-x^*|\leqslant\cdots\leqslant L^{k+1}|x_0-x^*|$

当$k\to\infty$，$|x_k-x^*|\to 0$，迭代序列$\{x_k\}$收敛到根x^*，实际上，对迭代序列（5.3.2）有收敛定理：

5.3.8 **定理** 假定$g(\cdot)$在$[a, b]$上可微，$|g'(x)|\leqslant L<1$，且当$x\in[a, b]$时$a\leqslant g(x)\leqslant b$，则对任何初始近似$x_0\in[a, b]$，由迭代（5.3.2）生成的序列$\{x_k\}$均收敛于方程（5.3.1）的根$x^*$，并有误差估计

5.3.9 $|x_k-x^*|\leqslant\dfrac{L^k}{1-L}|x_1-x_0|$

在例(5.3.4) 中，当 $g(x)=\sqrt[3]{1+x}$ 时 $g'(x)=\frac{1}{3}(1+x)^{-2/3}$，在区间[1,2]中，$\max\limits_{1\leqslant x\leqslant 2}|g'(x)|=\frac{1}{3}\frac{1}{\sqrt[3]{4}}<0.21<1$ 且当 $x\in[1,2]$ 时，$1\leqslant g(x)\leqslant 2$，故迭代 (5.3.5) 收敛。对迭代 (5.3.6)，$g'(x)=3x^2$，当 $x\in[1,2]$ 时，$g'(x)\geqslant 3$ 定理条件不满足，因此迭代 (5.3.6) 不能使用。定理(5.3.8) 既给出了迭代 (5.3.2) 收敛到根 x^* 的充分条件，也给出了方程 (5.3.1) 在 $[a,b]$ 上根的存在性，它是一个大范围收敛的定理，条件较强，不易满足，通常只在所求根 x^* 附近研究序列 $\{x_k\}$ 的收敛性及收敛速度。

5.3.10 **定义** 对任何 $x_0\in R$，R：$|x-x^*|\leqslant\delta,\delta>0$，迭代法 (5.3.2) 生成的序列 $\{x_k\}$ 均收敛到 x^*，则称此迭代序列具有**局部收敛性** (Local Convergence)。

5.3.11 **定理** 假定 x^* 是方程 (5.3.1) 的根，$g'(x)$ 在 x^* 的邻域连续，且 $|g'(x^*)|<1$，则迭代法 (5.3.2) 是局部收敛的。

5.3.12 **定义** 迭代序列 $\{x_k\}$ 收敛于 x^*，如果存在实数 $p\geqslant 1$ 和常数 $C\neq 0$，若当 $k\geqslant k_0$ 时 $x_k\neq x^*$，渐近关系式

$$\lim_{k\to\infty}\frac{x_{k+1}-x^*}{(x_k-x^*)^p}=C\neq 0$$

成立，则称迭代序列 $\{x_k\}$ 是 p **阶收敛**的 (Order p Convergence)。$p=1$ 称为**线性收敛** (Linear Convergence)，$p>1$ 称为**超线性收敛**(Superlinear Convergence)，$p=2$ 称为**平方收敛**(Quadratic Convergence)。

5.3.13 **定理** 设 x^* 是方程 (5.3.1) 的根，若 $g^{(p)}(x)$ 在根 x^* 附近连续，并且

$g'(x^*)=\cdots=g^{(p-1)}(x^*)=0$，$g^{(p)}(x^*)\neq 0$

则迭代 (5.3.2) 是 p 阶收敛的。

迭代法（5.3.2）的优点是计算简单，但通常它只有线性敛速甚至不收敛，为此需要改善迭代公式或构造具有较高敛速的迭代方法。

5.3.2 迭代法的加速收敛

如果迭代序列（5.3.2）收敛很慢，要达到要求的精度将使计算量很大，为此，需研究加速迭代收敛性的方法。

设x_k是根x^*的近似，$x_k-x^*\neq0$，由（5.3.7）可得

$$\frac{x_{k+1}-x^*}{x_k-x^*}=g'(\xi_k),\quad \xi_k\text{在}x_k\text{与}x^*\text{之间}$$

假定 $g'(x)$ 在x变化时改变不大，可令 $g'(x)\approx L$，于是可得

$$\frac{x_{k+1}-x^*}{x_k-x^*}\approx\frac{x_{k+2}-x^*}{x_{k+1}-x^*}$$

由此解出

5.3.14 $$x^*\approx x_{k+2}-\frac{(x_{k+2}-x_{k+1})^2}{x_{k+2}-2x_{k+1}+x_k}=x_{k+2}-\frac{(\Delta x_{k+1})^2}{\Delta^2x_k}=\overline{x}_{k+2}$$

其中$\Delta x_{k+1}=x_{k+2}-x_{k+1}$是$x_{k+1}$处的一阶差分，$\Delta^2x_k=\Delta x_{k+1}-\Delta x_k=x_{k+2}-2x_{k+1}+x_k$是$x_k$处的二阶差分。用$\overline{x}_{k+2}$近似$x^*$，就是用迭代法的误差对$x_{k+2}$进行校正，以加速收敛，而计算量只在原基础上增加计算一阶及二阶差分的算术运算。这种加速收敛的方法称为Aitken（埃特金）方法，也称δ^2过程。如果把由x_k到$\overline{x}_{k+2}$算做一步，则Aitken 方法的计算公式如下：

5.3.15 $$\begin{cases}\widetilde{x}_{k+1}=g(x_k)\\ \overline{x}_{k+1}=g(\widetilde{x}_{k+1})\\ x_{k+1}=\overline{x}_{k+1}-\dfrac{(\overline{x}_{k+1}-\widetilde{x}_{k+1})^2}{\overline{x}_{k+1}-2\widetilde{x}_{k+1}+x_k}\end{cases}\quad(k=0,1,\cdots)$$

它一步计算两个$g(x)$值，再进行校正。

5.3.16 **例** 用Aitken 方法求方程

$$x=e^{-x}$$

在$x_0=0.5$附近的根。

解 若用迭代法$x_{k+1}=e^{-x_k}$求根，计算18步可得$x_{18}=0.56714$。用Aitken方法，则$g(x_k)=e^{-x_k}$，代入公式（5.3.15），计算结果为：

$$x_0=0.5,\ \tilde{x}_1=0.60653,\ \bar{x}_1=0.54524,\ x_1=0.56762$$

$$\tilde{x}_2=0.56687,\ \bar{x}_2=0.56730,\ x_2=0.56714$$

这里只用2步（相当迭代法4步）就得到与迭代法x_{18}相同的结果，可见这方法对加速收敛效果较明显。甚至有些不收敛的迭代法用Aitken 方法处理也可得到收敛序列，如例(5.3.4)中的迭代公式（5.3.6），当$x_0=1.5$时迭代不收敛，但改用Aitken方法（5.3.15），只要计算5步就能得到$x_5=1.32472$，精度达到10^{-5}。

5.4

5.4 Newton法

Newton（牛顿）法是通过非线性方程线性化得到迭代序列的一种方法。

对于方程

5.4.1 $f(x)=0$

若已知根x^*的一个近似值x_k，可将$f(\cdot)$在x_k处展成一阶Taylor（泰勒）公式

$$f(x)=f(x_k)+f'(x_k)(x-x_k)+\frac{f''(\xi)}{2!}(x-x_k)^2$$

取其线性部分近似，即用线性方程

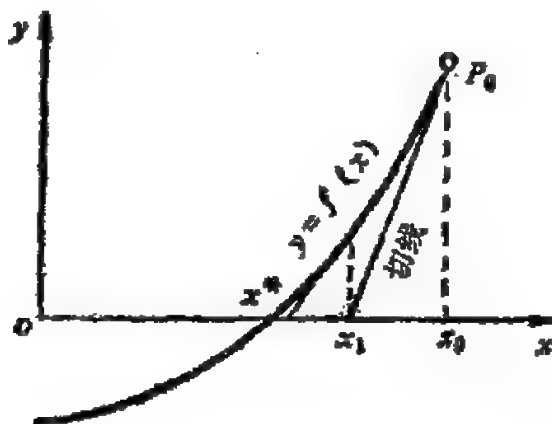
5.4.2 $f(x_k)+f'(x_k)(x-x_k)=0$

近似方程（5.4.1）。若$f'(x_k)\neq 0$，方程（5.4.2）的根记作x_{k+1}，则得

5.4.3 $x_{k+1}=x_k-\dfrac{f(x_k)}{f'(x_k)} \quad (k=0,1,\cdots)$

这就是Newton 法的迭代程序，它实际上是在点x_k处作曲线$y=f(x)$的切线，并用切线与x轴交点x_{k+1}做为根x^*的新近似，如（5.4.4）所示，故Newton 法也称**切线法**。

5.4.4 图



从迭代程序（5.4.3)可知，迭代函数为

$$g(x)=x-\frac{f(x)}{f'(x)}$$

于是

$$g'(x)=1-\frac{f'(x)}{f'(x)}+\frac{f(x)f''(x)}{[f'(x)]^2}=\frac{f(x)f''(x)}{[f'(x)]^2}$$

若$f(x^*)=0$，$f'(x^*)\neq0$，则$g'(x^*)=0$，且$g''(x^*)\neq0$。由定理（5.3.13)可知，Newton法生成的迭代序列$\{x_k\}$在x^*附近平方收敛，说明Newton 法收敛很快。

5.4.5 **例** 用Newton 法求方程

$$f(x)=xe^x-1=0$$

在$x_0=0.5$附近的根。

解 此方程的Newton 迭代程序为

$$x_{k+1}=x_k-\frac{x_k-e^{-x_k}}{1+x_k} \quad (k=0,1,\cdots)$$

直接计算可得：

$x_0=0.5$，$x_1=0.57102$，$x_2=0.56716$，$x_3=0.56714$

只算3步就达到10^{-5}精度，可见Newton 法收敛很快。

将Newton法应用于$f(x)=x^2-c=0$，求得$\sqrt{c}$ 的Newton程序为

5.4.6 $$x_{k+1}=x_k-\frac{x_k^2-c}{2x_k}=\frac{1}{2}\left(x_k+\frac{c}{x_k}\right)\quad(k=0,1,\cdots)$$

如求$\sqrt{115}$，即$c=115$。取$x_0=10$，用公式（5.4.6）计算 3步则得$x_3=10.723805$，精度达到10^{-6}。每步只用一次除法和一次加法，计算量省，公式（5.4.6）可作为计算$\sqrt{x}$的标准子程序算法。

Newton 法具有收敛快、可用于求复根等优点，但是此法只具有局部收敛性，即初始近似x_0要在根x^*附近。为了扩大收敛范围，可以采用Newton下山程序

5.4.7 $$x_{k+1}=x_k-\lambda_k\frac{f(x_k)}{f'(x_k)}\quad(k=0,1,\cdots)$$

其中参数λ_k的选择，应满足下山条件：

5.4.8 $$|f(x_{k+1})|<|f(x_k)|$$

故λ_k称为**下山因子**。若令

$$\bar{x}_{k+1}=x_k-\frac{f(x_k)}{f'(x_k)}$$

则Newton 下山法（5.4.7）等价于

5.4.9 $$x_{k+1}=\lambda_k\bar{x}_{k+1}+(1-\lambda_k)x_k$$

使用Newton 下山法求根时，下山因子λ_k可用逐步搜索法确定，即先令$\lambda_k=1$，判断条件（5.4.8）是否成立，若不成立再将λ_k缩小$\frac{1}{2}$，直到条件（5.4.8）成立为止。

5.4.10 例 用Newton 法求方程

$$f(x)=x^3-x-1=0$$

解 计算程序

5.4.11 $x_{k+1}=x_k-\dfrac{x_k^3-x_k-1}{3x_k^2-1}\quad(k=0,1,\cdots)$

当$x_0=1.5$时，计算3步得$x_3=1.32472$，因为x_0与x^*很靠近，故收敛很快。但如取$x_0=0.6$则由程序（5.4.11）求得$x_1=17.9$，再算下去显然不会收敛。如用Newton下山法(5.4.9)。令$\overline{x}_1=17.9$，从$\lambda_0=1$开始逐次搜索，当$\lambda_0=\dfrac{1}{32}$时由（5.4.9）可得

$$x_1=\frac{1}{32}\overline{x}_1+\frac{31}{32}x_0=1.140625$$

满足条件$|f(x_1)|<|f(x_0)|$，x_1已修正了$\overline{x}_1$的严重偏差，以后计算由于$\lambda_k=1$就能使条件（5.4.8）成立。因此Newton下山法与Newton法结果一样，$x_2=\overline{x}_2=1.366814$，$x_3=\overline{x}_3=1.32628$，$x_4=\overline{x}_4=1.32472$。

5.5

5.5 弦截法与抛物线法

Newton法是用切线近似曲线$y=f(x)$求得新近似值，它要计算导数$f'(x_k)$。当计算$f'(x)$较困难时，往往用$f(x)$在一些点上的函数值来近似，如用曲线上两点确定的直线来近似曲线求得方程的近似根。这种方法称为弦截法。如用曲线上三个点做抛物线近似曲线，求得方程新的近似根的方法称为抛物线法。更一般的就是用$y=f(x)$的插值多项式$P_n(x)$近似$f(x)$求方程的根，这就是插值求根法。

5.5.1 弦截法

设曲线$y=f(x)$上两点$(x_0,\ f(x_0))$，$(x_1,\ f(x_1))$已知，通过这两点的直线方程为

5.5.1 $P_1(x)=f(x_1)+\frac{f(x_1)-f(x_0)}{x_1-x_0}(x-x_1)$

若$f(x_1)\neq f(x_0)$，则直线$y=P_1(x)$与x轴（$y=0$)的交点为$(x_2,\ 0)$，其中

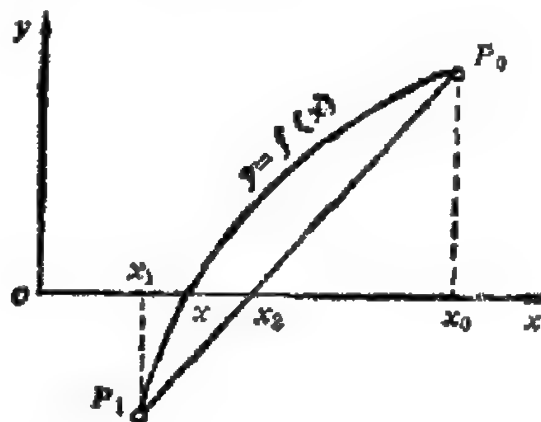
$$x_2=x_1-\frac{x_1-x_0}{f(x_1)-f(x_0)}f(x_1)$$

把x_2作为新的近似，再由$(x_2,\ f(x_2))$与$(x_1,\ f(x_1))$两点定出x_3，依此类推，若两点$(x_k,\ f(x_k))$与$(x_{k-1},\ f(x_{k-1}))$已知，在$f(x_k)\neq f(x_{k-1})$时，有

5.5.2 $x_{k+1}=x_k-\frac{x_k-x_{k-1}}{f(x_k)-f(x_{k-1})}f(x_k)\quad (k=1,2,\cdots)$

这就是弦截法程序．它相当于Newton 法（5.4.3）中导数$f'(x_k)$用x_{k-1}与x_k的差商来代替．

5.5.3 图



5.5.4 例 用弦截法求方程

$f(x)=xe^x-1=0$

在$x_0=0.5$附近的根．

解 取$x_0=0.5$，$x_1=0.6$，用弦截法公式(5.5.2)计算，每步算一次函数值，可求得：

$x_2=0.56532$，$x_3=0.56709$，$x_4=0.56714$．

例(5.5.4)的结果与例(5.4.4)的Newton法结果比较，可以看出弦截法的收敛速度也相当快．它有以下的局部收敛定理．

5.5.5 定理 假定$f(\cdot)$在根x^*的邻域Δ:$|x-x^*|<\delta$内具有二阶连续导数，且对任意$x\in\Delta$有$f'(x)\neq0$，初始近似$x_0,x_1\in\Delta$，则当Δ充分小时，弦截法（5.5.2)生成的序列$\{x_k\}$收敛到x^*，

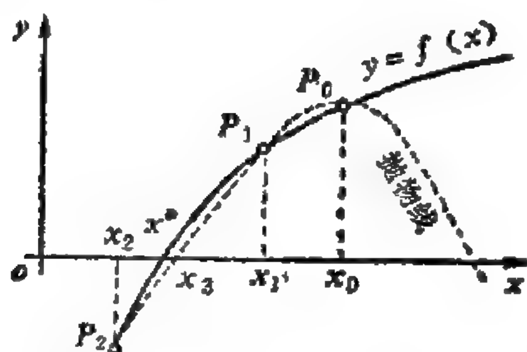
且收敛阶为 $p=\dfrac{1+\sqrt{5}}{2}\approx1.618$.

弦截法收敛阶虽比Newton 法低，但它不用计算导数，且每步只算一次函数值，计算量少，因此，它是一个效率较高、适于在计算机上求方程根的方法。

5.5.2 抛物线法

给定曲线$y=f(x)$上三个不共线的点$(x_0, f(x_0))$，$(x_1, f(x_1))$，$(x_2, f(x_2))$，通过这三点作抛物线$y=P_2(x)$近似$y=f(x)$，适当选取$P_2(x)=0$的一个根记作x_3，做为方程的新近似根。这样确定的迭代过程就是抛物线法，也称Muller（密勒)法，几何图形就是用抛物线$y=P_2(x)$与x轴的交点横坐标x_3作为根x^*的近似，如图（5.5.6）所示。

5.5.6 图



已知$(x_k,f(x_k))$，$(x_{k-1},f(x_{k-1}))$，$(x_{k-2},f(x_{k-2}))$三点不共线，由Newton 插值多项式有

$$\begin{aligned}P_2(x)&=f(x_k)+f[x_k,x_{k-1}](x-x_k)\\&\quad+f[x_k,x_{k-1},x_{k-2}](x-x_k)(x-x_{k-1})\\&=f[x_k,x_{k-1},x_{k-2}](x-x_k)^2+(f[x_k,x_{k-1}]\\&\quad+f[x_k,x_{k-1},x_{k-2}](x_k-x_{k-1}))\cdot(x-x_k)+f(x_k)\end{aligned}$$

由于$f[x_k,x_{k-1},x_{k-2}]\neq0$，故$P_2(x)=0$有二个根

5.5.7 $$x-x_k=\frac{-w\pm(w^2-4f(x_k)f[x_k,x_{k-1},x_{k-2}])^{1/2}}{2f[x_k,x_{k-1},x_{k-2}]}$$

其中

5.5.8 $$w=f(x_k,x_{k-1})+f[x_k,x_{k-1},x_{k-2}](x_k-x_{k-1})$$

为避免有效位数损失，对(5.5.7) 采用有理化分子，并把x记作x_{k+1}，可得抛物线法计算公式

5.5.9 $$x_{k+1}=x_k-\frac{2f(x_k)}{w\pm(w^2-4f(x_k)f(x_k,x_{k-1},x_{k-2}))^{1/2}}$$

$$(k=2,3,\cdots)$$

其中w由 (5.5.8) 给出．在两根中应选与x_k靠近的值作为新近似根x_{k+1}，为此，取根式前的“$\pm$”号要与w同号．

5.5.10 **例** 用抛物线法求方程

$f(x)=xe^x-1=0$

在$x_0=0.5$附近的根．

解 取$x_0=0.5, x_1=0.6, x_2=0.56532$为初始近似，计算相应函数值与差商值：

$f(x_0)=-0.175639$, $f(x_1)=-0.093271$

$f(x_2)=-0.005031$, $f[x_1,x_0]=2.68910$

$f[x_2,x_1]=2.83373$, $f[x_2, x_1, x_0]=2.75694$

代入 (5.5.8) 算出 $w=2.75691$，再由 (5.5.9) 求得 $x_3=0.56714$．此例精度已达到10^{-5}，说明抛物线法比弦截法收敛更快．事实上，在一定条件下可证明抛物线法的收敛阶$p\approx 1.840$．抛物线法具有收敛快，能算复根等优点，缺点是要用开方运算，计算公式较复杂，因此在计算机上使用时，计算公式应做适当改变．其计算步骤如下：

1° 准备．选定初始近似x_0, x_1, x_2，并计算相应的 f_0, f_1, f_2，及 $\lambda_2=\frac{x_2-x_1}{x_1-x_0}$．

2° 迭代．计算

$\delta_2=1+\lambda_2$

$a=(\lambda_2 f_0-\delta_2 f_1+f_2)\lambda_2$

$b=\lambda_2^2 f_0-\delta_2^2 f_1+(\lambda_2+\delta_2)f_2$

$c=\delta_2 f_2$

$$\lambda_3=\frac{-2c}{b\pm\sqrt{b^2-4ac}}$$

上式分母中"±"号与b取同号，于是得到$x_3=x_2+\lambda_3(x_2-x_1)$，再计算$f_3=f(x_3)$。

3° 控制。如果x_3满足$|\delta|\leqslant\varepsilon_1$或$|f_3|\leqslant\varepsilon_2$($\varepsilon_1$，$\varepsilon_2$为给定精度)，则认为迭代收敛，终止迭代，$x_3$即为所求，否则执行4°。这里

$$\delta=\begin{cases}|x_3-x_2|, & 当\ |x_3|<\gamma\\ \dfrac{|x_3-x_2|}{|x_3|}, & 当\ |x_3|\geqslant\gamma\end{cases}\qquad \gamma为控制常数$$

4° 修改。如迭代次数达到指定的次数N，则认为迭代不收敛，否则以(x_1，x_2，x_3，f_1，f_2，f_3，λ_3)分别代替(x_0，x_1，x_2，f_0，f_1，f_2，λ_2)，转2°继续迭代。

5.6

5.6 代数方程求根问题

当$f(x)$是由(5.1.2)给出的代数多项式时，方程(5.1.1)就称为代数方程，由于多项式有很多特殊性质，要求不同：有时只要知道根的界，有时只要判断右半平面有没有根，有时则要求全部根，因此，代数方程求根也有很多特殊方法。

5.6.1 多项式求值与Newton法

在各种求根方法中，计算工作量主要指求$f(x)$值的多少。如用Newton 法求根，每步要计算$f(x_k)$及$f'(x_k)$各一次。对多项式方程

$$f(x)=a_0x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0 \tag{5.6.1}$$

为了计算$f(x_k)$的值，可用$x-x_k$除$f(x)$得

5.6.2 $f(x)=q(x)(x-x_k)+f(x_k)$

其余式$f(x_k)$就是多项式$f(x)$ 在点x_k的值.商$q(x)$ 可表示为
$q(x)=b_0x^{n-1}+\cdots+b_{n-2}x+b_{n-1}$

比较（5.6.2）两端x同次幂系数，则得

5.6.3
$$\begin{cases} b_0=a_0 \\ b_i=a_i+x_kb_{i-1} & (i=1,\cdots n) \\ f(x_k)=b_n \end{cases}$$

用这个公式计算多项式值的方法称为秦九韶方法，它只需用n次乘法和n次加法运算，因此计算量省，结构紧凑，便于编制程序。应用这个方法求$f'(x_k)$，只要对（5.6.2）两端求导，则可得$f'(x_k)=q(x_k)$，也就是对$q(x)$继续用秦九韶方法。令

5.6.4 $q(x)=p(x)(x-x_k)+q(x_k)$

$p(x)=c_0x^{n-2}+\cdots+c_{n-3}x+c_{n-2}$

比较（5.6.4）两端系数则得

5.6.5
$$\begin{cases} c_0=b_0 \\ c_i=b_i+x_kc_{i-1} & (i=1,\cdots,n-1) \\ f'(x_k)=q(x_k)=c_{n-1} \end{cases}$$

根据以上算法，用Newton 法求方程（5.6.1）根的计算公式是

5.6.6 $x_{k+1}=x_k-\dfrac{f(x_k)}{f'(x_k)}=x_k-\dfrac{b_n}{c_{n-1}}$ （$k=0,1,\cdots$）

由x_k算出x_{k+1}的步骤是：

1° 令$b_0=c_0=a_0$；

2° 对$i=1$，…，$n-1$做

$b_i=a_i+x_kb_{i-1}$，$c_i=b_i+x_kc_{i-1}$；

3° 由$b_n=a_n+x_kb_{n-1}$，得$x_{k+1}=x_k-\dfrac{b_n}{c_{n-1}}$。

整个求解过程只要对$k=0,1,\cdots$，算到x_{k+1}满足精度要求为止。

5.6.2 根模的上下界

有的问题只要求知道根模的界，如特征值估计；而知道根模的界也就得到有根区域。为了估计根模的界，可把方程（5.6.1）改写成标准形式

5.6.7 $f(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0$

它的n个根x_i（$i=1, \cdots, n$)的绝对值$|x_i|$称为根模。如有常数$0<A<B$，使

$A\leqslant|x_i|\leqslant B \quad (i=1, \cdots, n)$

成立，则称A为根模下界，B为根模上界。求根模上界有以下定理。

5.6.8 **定理** 对方程（5.6.7）的所有根x_i，有

$$|x_i|\leqslant\max\{|a_1|+1,\cdots, |a_{n-1}|+1, |a_n|+1\}$$

或

$$|x_i|\leqslant\max\left\{1, \sum_{j=1}^{n}|a_j|\right\}$$

这种求根模的方法很简单，但求出的上界通常太大。为求得更好的上界，可令

$u(\rho)=\rho^n-|a_1|\rho^{n-1}-\cdots-|a_{n-1}|\rho-|a_n|$

显然$|f(x)|\geqslant u(|x|)$。由于$u(0)=-|a_n|$，$u(+\infty)>0$，方程$u(\rho)=0$在$(0, +\infty)$有唯一正根ρ_0，所以当$\rho\in(\rho_0, +\infty)$时$u(\rho)>0$。若存在ρ_1，使$u(\rho_1)>0$，则当$|x|\geqslant\rho_1$时$|f(x)|\geqslant u(|x|)>0$，这时方程（5.6.7）无根，故ρ_1是根模的一个上界。

5.6.9 **例** 估计方程$f(x)=x^3-2x-5=0$的根模的上界。

解 由定理(5.6.8)有$|x_i|\leqslant6$，6是根模一个上界。为了求更好的上界，可取$\rho_1=3$，则$u(3)=16>0$，故$|x_i|\leqslant3$，3是一个更好上界。若取$\rho_1=2.1$，因$u(2.1)=0.061>0$，故$|x_i|\leqslant2.1$是一个更好的根模上界。

为了求根模下界，可令

$$L(\sigma)=\sigma^n+|a_1|\sigma^{n-1}+\cdots+|a_{n-1}|\sigma-|a_n|$$

显然有 $|f(x)|\geq -L(|x|)$。当$\sigma>0$，$L(\sigma)$是单调增函数时，$L(0)<0$，$L(+\infty)>0$，故$L(\sigma)$只有一个正根$\sigma_0>0$。当 $0\leq\sigma<\sigma_0$时$L(\sigma)<0$，若存在$\sigma_1<\sigma_0$，使 $L(\sigma_1)<0$，则当$|x|\leq\sigma_1$，有$L(|x|)<0$。因此$|f(x)|\geq -L(|x|)>0$，表明$|x|\leq\sigma_1$时方程（5.6.7）没有根，故$|x_i|\geq\sigma_1$，σ_1即是根模的一个下界，而根模$|x_i|$的最大下界为σ_0。

如果$f(x)\neq 0$，令$z=\dfrac{1}{x}$，则z满足另一个 n 次代数方程

$$a_nz^n+a_{n-1}z^{n-1}+\cdots+a_1z+1=0$$

若$a_n\neq 0$，以a_n除之即得

$$g(z)=z^n+\frac{a_{n-1}}{a_n}z^{n-1}+\cdots+\frac{a_1}{a_n}z+\frac{1}{a_n}=0$$

对$g(z)=0$应用定理（5.6.8）的结论，则得

$$|z_i|=\left|\frac{1}{x_i}\right|\leq\max\left\{\left|\frac{a_{n-1}}{a_n}\right|+1,\cdots,\left|\frac{a_1}{a_n}\right|+1,\left|\frac{1}{a_n}\right|+1\right\}$$

$$=b\geq 1$$

$$|z_i|=\left|\frac{1}{x_i}\right|\leq\max\left\{1,\sum_{j=1}^{n-1}\left|\frac{a_j}{a_n}\right|+\left|\frac{1}{a_n}\right|\right\}=c\geq 1$$

于是有根模下界$|x_i|\geq\dfrac{1}{b}$或$|x_i|\geq\dfrac{1}{c}$。

5.6.10 例 估计方程$f(x)=x^3-2x-5=0$的根模的下界。

解 由于

$$b=\frac{2}{5}+1=\frac{7}{5}，c=1$$

故$\frac{5}{7}$是根模的一个下界。1是一个更好的下界，若取$\sigma=1.3$，

则$L(1.3)=-0.203<0$，故1.3是一个更好的根模下界。与例(5.6.9)结果合在一起，可知$f(x)=x^3-2x-5=0$的根均在下列圆环内，即$1.3\leqslant|x_i|\leqslant2.1$。

5.6.11 例 估计方程$f(x)=x^{41}+x^3+1=0$的根模上、下界。

解 若用定理(5.6.8)可估得根模上界为2，下界$\frac{1}{2}$，即

$\frac{1}{2}\leqslant|x_i|\leqslant2$。若取$\rho_1=1.018$，计算函数值$u(1.018)=(1.018)^{41}-(1.018)^3-1=0.023>0$，故1.018是一个更好上界。为估计下界，考虑$L(\sigma)=\sigma^{41}+\sigma^3-1$，取$\sigma_1=0.9524$，计算$L(0.9524)=0.9993-1=-0.0007<0$，故0.9524是一个更好的下界。于是得到这个方程41个根都在以原点为中心的下列圆环内：

$0.9524\leqslant|x_i|\leqslant1.018$

这是单位圆邻近很狭窄的一个圆环域。

5.6.3 Sturm 序列

为了解决代数方程根的隔离问题，引进以下定义。

5.6.12 定义 实多项式序列

5.6.13 $f(x)=f_0(x),\ f_1(x),\ \cdots,\ f_m(x)$

称为Sturm（斯特姆）序列，如果

1° ξ是$f_k(x)$的实零点，则对$k=1,\ \cdots,\ m-1$，有$f_{k-1}(\xi)f_{k+1}(\xi)<0$

2° 最后一个多项式$f_m(x)$没有实根。

设$f_0(x)$和$f_1(x)$是两个x的多项式，其中$f_1(x)$次数低于$f_0(x)$。用$f_1(x)$除$f_0(x)$得商$q_0(x)$及余式$R_2(x)$，$R_2(x)$的次数低于$f_1(x)$。令$f_2(x)=-R_2(x)$为序列的下一个函数，如

果$f_2(x)\neq 0$，用$f_2(x)$除$f_1(x)$得商$q_1(x)$及余式$R_3(x)$，$R_3(x)$次数低于$f_2(x)$。令$f_3(x)=-R_3(x)$为下一个函数，这样做下去，直到除尽为止。从f_0，f_1得出$m+1$个非零多项式$\{f_0, f_1, \cdots, f_m\}$，次数逐个降低，$f_{m+1}\equiv 0$，这个函数序列中三个相邻函数间的关系是

$$f_k(x)=q_k(x)f_{k+1}(x)-f_{k+2}(x)\quad (k=0,\ 1,\ \cdots,\ m-1)$$

$$f_{m+1}(x)\equiv 0$$

这样得到的序列就是一个Sturm 序列，称为以f_0及f_1为基的Sturm 序列。通常选$f_0=f$，$f_1=f'(x)$为基作Sturm序列

5.6.14 $\{f,\ f',\ f_2\ ,\cdots,\ f_m\}$

5.6.15 **定义** 实多项式序列（5.6.13），当$x=a$时是一个数列$\{f_0(a),\ f_1(a),\ \cdots,\ f_m(a)\}$，若两个相邻数符号相反，就说这两个数之间有一次变号，把数列中一切零拿出数列，则数列中各相邻数变号次数之和定义为这个数列的**变号次数**(The Number of Changes in Sign)。记作V_a。

5.6.16 **定理/Sturm** 设$f(a)\neq 0$，$f(b)\neq 0$，以$f_0=f$，$f_1=f'(\cdot)$为基的Sturm序列（5.6.14）在点x的变号次数为V_x，则方程$f(\cdot)=0$在$(a,\ b)$区间内共有V_a-V_b个各不相同的实根，设最后非零函数$f_m(x)$没有实根，则$f(x)=0$的实根都是单根；设$f_m(x)=0$有实根,则这些根都是$f(x)=0$的重根，其重数为$f_m(x)=0$根的重数加1。

Sturm 定理解决了判断给定区间内有没有根，有几个实根的问题。当$a=-\infty$，$b=+\infty$时，可解决$f(x)=0$的实根个数；当$(a,\ b)$内只有一个根，且$b-a$很小时，可解决实根隔离，用二分法容易求出根的足够精确的近似值。

5.6.17 **例** $f(x)=2x^3-9x^2+11x-3.5=0$

解 Sturm 序列是

$$f_0(x)=f(x)=2x^3-9x^2+11x-3.5$$

$$f_1(x)=f'(x)=6x^2-18x+11$$

$f_2(x)=\frac{1}{3}(5x-6)$

$f_3(x)=1$

由于求$f_k(x)$时可以相差一个正常数因子，这里$f_3(x)=1$，故没有重根，相应点变号次数结果如下：

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	V_x
$-\infty$	−	+	−	+	3
0	−	+	−	+	3
$+\infty$	+	+	+	+	0

可见方程只有三个正实根，没有负根。为进一步将这三个根隔离，可再试算某些点的变号次数，结果如下：

x	$f_0(x)$	$f_1(x)$	$f_2(x)$	$f_3(x)$	V_x
0	−	+	−	+	3
1	+	−	−	+	2
2	−	−	+	+	1
3	+	+	+	+	0

这说明三个根分别在区间(0，1)，(1，2)，(2,3)中，可用二分法或其它精确化方法把根计算出来。

5.7

5.7 Bernoulli方法

对任何n次方程

5.7.1 $f(x)=x^n+a_1x^{n-1}+\cdots+a_{n-1}x+a_n=0$

考虑其对应的齐次常系数线性差分方程

5.7.2 $f(E)u_k = u_{k+n} + a_1 u_{k+n-1} + \cdots + a_{n-1}u_{k+1} + a_n u_k = 0$

它的特征方程就是n次方程（5.7.1），方程（5.7.1）模最大的根叫最大根，模最小的根叫最小根。Bernoulli（伯努利）方法通常用来求最大根，稍加变动也可用来求方程的最小根。

设方程的最大根x_1，其它根x_2，…，x_n的模都比x_1的模小。假定给出差分方程的初始条件$u_0 = \cdots = u_{n-2} = 0$，$u_{n-1} = 1$，把差分方程（5.7.2）改写成

5.7.3 $u_{k+n} = -a_1 u_{k+n-1} - \cdots - a_{n-1}u_{k+1} - a_n u_k$

由此可逐次算出它的特解$u_n = -a_1$，…。

这个特解可表示成

5.7.4 $u_k = c_1 x_1^k + c_2 x_2^k + \cdots + c_n x_n^k$，$c_1 \neq 0$

这里c_i是由初始条件确定的。把（5.7.4）改写成

$$u_k = c_1 x_1^k \left[1 + \frac{c_2}{c_1}\left(\frac{x_2}{x_1}\right)^k + \cdots + \frac{c_n}{c_1}\left(\frac{x_n}{x_1}\right)^k\right]$$

于是有

$$\frac{u_{k+1}}{u_k} = \frac{c_1 x_1^{k+1}\left[1 + \sum_{i=2}^{n}\left(\frac{c_i}{c_1}\right)\left(\frac{x_i}{x_1}\right)^{k+1}\right]}{c_1 x_1^k\left[1 + \sum_{i=2}^{n}\left(\frac{c_i}{c_1}\right)\left(\frac{x_i}{x_1}\right)^k\right]}$$

由于$|x_1| > |x_2| \geqslant \cdots \geqslant |x_n|$，故

$$\lim_{k\to\infty}\frac{u_{k+1}}{u_k} = x_1$$

这表明：当k充分大时，后面的数u_{k+1}与前面的数u_k的比值是最大根x_1的近似值。以上运算的方法就是求最大根的Bernoulli方法。当方程最大根是实单根或是一个复根（非一对共轭虚根）时，可用Bernoulli法。但是当$|x_2|$很接近$|x_1|$时，

其收敛速度很慢，Bernoulli 法就没有多大实用价值．计算(5.7.3)特解的方法编成程序后计算很简单，手算时可用计算样板，将方程系数从下到上排列，次序是$-a_1,\cdots,-a_n$，与相应u_k相乘。

5.7.5 例 求$f(x)=x^3+15x^2+69x+104=0$的最大根.

解 计算结果如下表：

	u_k	u_{k+1}/u_k
−104	0	
−69	0	
−15	1	−15
→	−15	−10.4
	156	−9.032
	−1409	−8.4677
	11931	−8.2104
	−97958	−8.0919
	792667	−8.03834
	−6371727	−8.014705
	51067514	−8.005107
	−408800915	−8.001481
	3271012787	

可以看出，当k增大时比值 u_{k+1}/u_k越来越接近最大实根-8．由于$f(x)=x^3+15x^2+69x+104=0$可以改写成 $f(x)=(x+8)(x^2+7x+13)=0$，方程最大根是-8，其它两个根是一对共轭虚根$\dfrac{-7\pm\sqrt{3}\,i}{2}$，它们的模是$\sqrt{13}\approx3.606$，$\dfrac{\sqrt{13}}{8}\approx0.45$，故收敛速度较慢。

这种方法也可用来求最小根．设方程(5.7.1)最小根$x_n\neq0$，则 $a_n\neq0$．令 $z=\dfrac{1}{x}$，代入方程(5.7.1)可得z满足

方程

5.7.6 $a_n z^n + a_{n-1} z^{n-1} + \cdots + a_1 z + 1 = 0$

只要对方程（5.7.6）求最大根．则得方程（5.7.1）的最小根．求最小根的Bernoulli 法也可作为根的精确化的一种 有效方法．

5.8

5.8 劈因子法

因为二次方程容易求解，故若能找出n次多项式 $f(x)$ 的一个二次因式，就等于找到了方程（5.6.1）的一对复根.劈因子法的基本思想就是从某个近似的二次因子

$w(x) = x^2 + ux + v$

出发，用某种迭代过程使之逐步逼近$f(x)$的一个二次因子，从而达到求根目的。

用$w(x)$除$f(x)$得商式$p(x)$，它是$n-2$次多项式.一次余式为

$r(x) = r_1 x + r_2$

其中r_1和r_2两个系数均由$w(x)$的系数u，v所确定，因此它们是u，v的函数，于是有

5.8.1 $f(x) = (x^2 + ux + v)p(x) + r_1 x + r_2$

若$r_1 = r_2 = 0$，则$w(x)$就是$f(x)$的准确二次因子，一般情形是$(r_1, r_2) \neq (0, 0)$。这时，可修正$w(x)$的系数u, v，修正值记作Δu与Δv，修正后的新二次因式为

$w^*(x) = x^2 + (u + \Delta u)x + (v + \Delta v)$

它比旧的二次因式$w(x)$更接近$f(x)$的对应的准确二次因子。用$w^*(x)$除$f(x)$得余式

$r^*(x) = (r_1 + \Delta r_1)x + (r_2 + \Delta r_2)$

若要求$r^*(x) = 0$，即要求Δu与Δv引起的Δr_1和Δr_2满足下列方程组：

5.8.2 $r_1 + \Delta r_1 = 0$，$r_2 + \Delta r_2 = 0$

若用微分近似增量，则得

$$
\begin{cases}
\dfrac{\partial r_1}{\partial u}\Delta u+\dfrac{\partial r_1}{\partial v}\Delta v+r_1=0 \\
\dfrac{\partial r_2}{\partial u}\Delta u+\dfrac{\partial r_2}{\partial v}\Delta v+r_2=0
\end{cases}
\tag{5.8.3}
$$

只要求出方程系数$\dfrac{\partial r_1}{\partial u}$、$\dfrac{\partial r_2}{\partial u}$、$\dfrac{\partial r_1}{\partial v}$、$\dfrac{\partial r_2}{\partial v}$及$r_1$，$r_2$，就可从(5.8.3)解出$\Delta u$及$\Delta v$，从而求得新的二次因子$w^*(x)$.计算步骤如下：

1° 计算r_1及r_2.令

$p(x)=b_0x^{n-2}+b_1x^{n-3}+\cdots+b_{n-2}$

代入(5.8.1)，比较同次幂系数，得

$$
\begin{cases}
b_0=a_0, \quad b_1=a_1-ub_0 \\
b_i=a_i-ub_{i-1}-vb_{i-2} \qquad (i=2,\cdots,n) \\
r_1=b_{n-1} \\
r_2=b_n+ub_{n-1}
\end{cases}
\tag{5.8.4}
$$

2° 计算$\dfrac{\partial r_1}{\partial u}$，$\dfrac{\partial r_1}{\partial v}$，$\dfrac{\partial r_2}{\partial u}$，$\dfrac{\partial r_2}{\partial v}$.将(5.8.1)两端分别对$u$，对$v$微分，可得

$$
\begin{cases}
xp(x)=-(x^2+ux+v)\dfrac{\partial p}{\partial u}-\dfrac{\partial r_1}{\partial u}x-\dfrac{\partial r_2}{\partial u} \\
p(x)=-(x^2+ux+v)\dfrac{\partial p}{\partial v}-\dfrac{\partial r_1}{\partial v}x-\dfrac{\partial r_2}{\partial v}
\end{cases}
\tag{5.8.5}
$$

由此式可知，$\dfrac{\partial r_1}{\partial u}$，$\dfrac{\partial r_2}{\partial u}$是以$w(x)$除$xp(x)$所得余式的系数，而$\dfrac{\partial r_1}{\partial v}$和$\dfrac{\partial r_2}{\partial v}$是以$w(x)$除$p(x)$所得余式的系数．若令

5.8.6 $p(x)=(x^2+ux+v)H(x)+s_1x+s_2$

其中

$H(x)=c_0x^{n-4}+c_1x^{n-5}+\cdots+c_{n-4}$

于是

5.8.7 $xp(x)=(x^2+ux+v)xH(x)+s_1x^2+s_2x$

$$=(x^2+ux+v)(xH(x)+s_1)-(us_1-s_2)x-vs_1$$

将（5.8.6），（5.8.7）与（5.8.5）比较，可得

5.8.8 $$\begin{cases} -\dfrac{\partial r_1}{\partial v}=s_1, \quad -\dfrac{\partial r_2}{\partial v}=s_2 \\ \dfrac{\partial r_1}{\partial u}=us_1-s_2, \quad \dfrac{\partial r_2}{\partial u}=vs_1 \end{cases}$$

为求得s_1及s_2，可通过（5.8.6）两端比较系数，得到

5.8.9 $$\begin{cases} c_0=b_0, \quad c_1=b_1-uc_0 \\ c_i=b_i-uc_{i-1}-vc_{i-2} \quad (i=2,\cdots,n-2) \\ s_1=c_{n-3} \\ s_2=c_{n-2}+uc_{n-3} \end{cases}$$

将求出的s_1及s_2代入（5.8.8），则得方程组（5.8.3）的系数．由于计算公式（5.8.9）与（5.8.4）相似，故编制的程序简单．

3° 由（5.8.3）解出Δu及Δv，若$|\Delta u|\leqslant\varepsilon$，$|\Delta v|\leqslant\varepsilon$成立，则迭代停止，转4°，否则令$u:=u+\Delta u, v:=v+\Delta v$，再转1°重新计算．

4° 求$w^*(x)=x^2+ux+v$的解

5.8.10 $x=\dfrac{-u\pm\sqrt{u^2-4v}}{2}$

上述步骤描述的算法叫做劈因子法，也称Bairstow（贝尔斯多夫）方法．对有一对共轭虚根的二次因式，它的收敛性与收敛速度与求复根的Newton 法等价，但它不用进行复

数运算。

5.8.11 **例** 用劈因子法求 $f(x)=x^4+x^3+5x^2+4x+4=0$的近似根。

解 取尾部作为近似二次因式，即

$w_0(x)=x^2+0.8x+0.8$

按步1°、2°的（5.8.4）、（5.8.9）、(5.8.8)求出方程组系数及自由项，得到方程组

$$\begin{cases} -0.608+3.72\Delta u-0.6\Delta v=0 \\ -0.768+0.48\Delta u+3.24\Delta v=0 \end{cases}$$

由此解出：$\Delta u=0.19697$，$\Delta v=0.20786$。

迭代一次得到

$w_1(x)=x^2+0.99697x+1.00786$

由（5.8.10）求出近似解

$\tilde{x}=-0.49849\pm0.87142i$

准确的二次因子是

$w^*(x)=x^2+x+1$

解是$x^*=-0.5\pm0.866i$.

若要得到更精确的二次因子，可从$w_1(x)$出发按同样方法再迭代下去，直到满足精度要求为止。

另一种简单的劈因子法是用 $xw(x)$ 除 $f(x)$ 得二次余式 $R(x)$，于是有

5.8.12 $f(x)=xw(x)S(x)+R(x)$

其中$S(x)$是商式，余式

$R(x)=r_0x^2+r_1x+r_2$

设$r_0\neq0$，则$R(x)$可改写成

$$R(x)=r_0\left(x^2+\frac{r_1}{r_0}x+\frac{r_2}{r_0}\right)=r_0\tilde{w}(x)$$

其中

$$\widetilde{w}(x)=x^2+\frac{r_1}{r_0}x+\frac{r_2}{r_0}=x^2+\widetilde{u}x+\widetilde{v}$$

$$\widetilde{u}=\frac{r_1}{r_0},\qquad \widetilde{v}=\frac{r_2}{r_0}$$

于是由（5.8.12）有

$$f(x)=xw(x)S(x)+r_0\widetilde{w}(x)$$

设$f(x)=0$没有零根，那么$f(x)$有二次因式的必要充分条件是$w(x)=\widetilde{w}(x)$。把从$w(x)$求出$\widetilde{w}(x)$的过程当作一个迭代过程，如果这个迭代过程收敛，则极限二次式就是$f(x)$的二次因式，这种迭代法叫林士谔法．使用这种方法，两次迭代比一次劈因子法的工作量小，而效果大致与一次劈因子法相当，因此计算程序比劈因子法简单得多．缺点是这个方法并不是都收敛的．

5.9

5.9 复根的隔离

使用Ruth（卢斯）定理，可以解决判断代数方程在右半平面有没有根，有几个根的问题．设给定n次实系数方程为

5.9.1 $$f(z)=z^n+a_1z^{n-1}+\cdots+a_{n-1}z+a_n=0$$

它有n个根，可能是实根，也可能是复根，表示为$z_1, z_2, \cdots, z_n$。如果有两个根相同就称为二重根．利用这些根，可把方程（5.9.1）改写成

5.9.2 $$f(z)=(z-z_1)\cdots(z-z_n)$$

设虚轴上没有方程（5.9.1）的根，则

$$f(it)\neq 0,\ it-z_k\neq 0(-\infty<t<+\infty)$$

由于复数乘积的幅角等于各因子幅角的和，所以有

$$\arg f(it)=\sum_{k=1}^{n}\arg(it-z_k)$$

当t从$+\infty$变到$-\infty$时，把t的角度函数$\phi(t)$的增量记作$\Delta\phi$，则

$$\Delta\phi=\phi(-\infty)-\phi(+\infty)$$

$$\Delta\arg f(it)=\sum_{k=1}^{n}\Delta\arg(it-z_k)$$

设根在右半平面上，则矢量$it-z_k$是从z_k到虚轴上动点it的矢量，当t从$+\infty$变到$-\infty$，它的角从$\frac{\pi}{2}$增加到$\frac{3}{2}\pi$

$$\Delta\arg(it-z_k)=\pi$$

设z_k在左半平面，则

$$\Delta\arg(it-z_k)=-\pi$$

设右半平面有r个根，左半平面有l个根，$l+r=n$，于是有

$$\frac{1}{\pi}\Delta\arg f(it)=\sum_{k=1}^{n}\frac{1}{\pi}\Delta\arg(it-z_k)$$

$$=r-l=r-(n-r)=2r-n$$

在右半平面根的个数为

5.9.3 $$r=\frac{1}{2}\left[n+\frac{1}{\pi}\Delta\arg f(it)\right]$$

为求r，记

$$\delta f(z)=\frac{1}{\pi}\Delta\arg f(it)$$

$$f(it)=i^n[f_0(t)-if_1(t)]=i^n w(t)$$

其中

5.9.4 $$\begin{cases}f_0(t)=t^n-a_2t^{n-2}+a_4t^{n-4}-\cdots\\ f_1(t)=a_1t^{n-1}-a_3t^{n-3}+a_5t^{n-5}-\cdots\end{cases}$$

$$w(t)=f_0(t)-if_1(t)$$

由于$\arg f(it)=\arg(i^n)+\arg w(t)$

i^n为常数，$\Delta i^n=0$，故

$$\delta f(z)=\frac{1}{\pi}\Delta\arg f(it)=\frac{1}{\pi}\Delta\arg w(t)$$

当t从$+\infty$变到$-\infty$时，研究$\phi(t)=\arg w(t)$的增量可改为研究$\operatorname{ctg}\phi(t)=-\dfrac{f_0(t)}{f_1(t)}$的变号情况．设$\operatorname{ctg}\phi$在虚轴上从正变到负共有$\alpha$次，从负变到正共有$\beta$次，则

$$\delta f(z)=\alpha-\beta$$

当$\operatorname{ctg}\phi$从正变到负时，$\{f_0(t),f_1(t)\}$从异号变为同号，损失一次变号；当$\operatorname{ctg}\phi$从负变到正时，$\{f_0(t),\ f_1(t)\}$增加一次变号．以$f_0(t)$，$f_1(t)$为基作Sturm序列

$$\{f_0(t),\ f_1(t),\ \cdots,\ f_m(t)\}$$

令

$$V_t=V\{f_0(t),\ f_1(t),\ \cdots,\ f_m(t)\}$$

则当t从$+\infty$变到$-\infty$时，Sturm 序列损失的变号次数是由$f_0(t)$的实零点引起的，中间函数的实零点对变号次数的改变无影响．由于已设$f(it)\neq 0$，所以$f_m(t)$没有实零点，它是

$$V_\infty-V_{-\infty}=\alpha-\beta=\delta f(z)$$

将其代入（5.9.3），则得

5.9.5 $\quad r=\dfrac{1}{2}(n+V_\infty-V_{-\infty})$

5.9.6 定理/Ruth 方程（5.9.1）在虚轴及右半平面没有根的充分必要条件是Sturm序列

$$\{f_0(t),\ f_1(t),\ \cdots,\ f_n(t)\}$$

内每个多项式比它前一个多项式低一次，且首项系数都是正数．

5.9.7 定理 设Sturm序列内有$n+1$个非零函数，则每个多项式比它的前一个多项式低一次，方程（5.9.1）在虚轴上没

有根，在右半平面根的个数等于首项系数组成的数列的变号次数。

5.9.8 定理 方程（5.9.1）在右半平面没有根，而在虚轴上有p个根的充分必要条件是Sturm 序列内有$n-p+1$个非零函数

$\{f_0(t),\ f_1(t),\ \cdots,\ f_{n-p}(t)\}$

其中每个多项式比它的前一个多项式低一次，首项系数都是正数，而且最后的p次多项式有p个实零点，这些实根就是方程（5.9.1）在虚轴上的p个根的虚部。

定理(5.9.6)、(5.9.7)、(5.9.8)给出了判断虚轴上及右半平面根的个数的方法，为了得到这些结论，要计算以（5.9.4）的f_0及f_1为基的Sturm 序列。由于f_0与f_1一个为偶函数，另一个是奇函数，故Sturm 序列中函数交替为奇偶函数，根据除法规则，除去零及交替正负号，可得到计算Sturm 序列系数的紧凑表格，叫做Ruth 表格。表（5.9.9）给出$n=6$的Ruth 表格：

5.9.9 表

f_0	1	a_2	a_4	a_6
f_1	a_1	a_3	a_5	
f_2	D_0	D_1	D_2	
f_3	E_0	E_1		
f_4	F_0	F_1		
f_5	G_0			
f_6	$H_0=F_1$			

其中

$$D_0=a_2-\frac{a_3}{a_1},\quad D_1=a_4-\frac{a_5}{a_1},\quad D_2=a_6,\cdots$$

计算系数的法则是：

5.9.10 本数＝上二行右数－$\dfrac{(上二行最左数)(上行右数)}{上行最左数}$

$f_k(t)$的实际系数是正负交替的，例如$f_2(t)=D_0t^4-D_1t^2+D_2$。如果首项系数都不等于零，根据定理（5.9.7），则此六次方程在右半平面的复根个数是

$V\{1, a_1, D_0, E_0, F_0, G_0, F_1\}$

另外，方程（5.9.1）在虚轴和右半平面没有根的必要条件是各系数都是正数，因此，如果方程有负系数或零系数，则此方程在虚轴或右半平面一定有根。

5.9.11 例 求$f(z)=z^4+z^2+z+1=0$

的最右根，即实部最大的根。

解 由于此方程z^3项系数为0，所以方程在虚轴或右半平面有根。容易验证此方程在虚轴上无根，故在右半平面有根。

把复平面原点移到$z_0=1$处，即做变换$u=z-1$，方程变成

$f(z)=u^4+4u^3+7u^2+7u+4=0$

为判断新的复平面右半平面有无根，即$\mathrm{Re}z>1$有无根，作Ruth表格：

1	7	4	$f_0(t)=t^4-7t^2+4$
4	7		$f_1(t)=4t^3-7t$
$\frac{21}{4}$	4		$f_2(t)=\frac{21}{4}t^2-4$
$\frac{83}{21}$			$f_3(t)=\frac{83}{21}t$
4			$f_4(t)=4$

因最左列系数均为正，根据定理（5.9.6），方程在$\mathrm{Re}z\geqslant1$没有根，故最右根范围在$0<\mathrm{Re}z<1$之间。选（0,1)中点$\frac{1}{2}$为复平面新原点，做$u=z-\frac{1}{2}$变换，方程变成

$$f(z)=u^4+2u^3+\frac{5}{2}u^2+\frac{5}{2}u+\frac{29}{16}=0$$

作Ruth表格：

$$\begin{array}{ccc} 1 & \frac{5}{2} & \frac{29}{16} \\ 2 & \frac{5}{2} & \\ \hline \frac{5}{4} & \frac{29}{16} & \\ -\frac{4}{10} & & \\ \frac{29}{16} & & \end{array}$$

最左列系数有两次变号，故在$\mathrm{Re}z>\frac{1}{2}$时有一对共轭复根，最右根范围缩小到$\frac{1}{2}<\mathrm{Re}z<1$，这样用对分法做下去，可求出最右根的实部近似值$x_0=0.5474$。求$f(z)$在$x_0$的展开式

$$f(z)=u^4+2.1896u^3+2.79788u^2+2.75091u+1.93684$$

作Ruth表格：

$$\begin{array}{ccc} 1 & 2.79788 & 1.93684 \\ 2.1896 & 2.75091 & \\ \hline 1.54153 & 1.93684 & \\ -0.00019 & & \\ 1.93684 & & \end{array}$$

$f_2(t)=1.54153t^2-1.93684$

$f_3(t)=-0.00019t\approx 0$

最左列系数仍有两次变号，但$f_3(t)\approx 0$，说明

$$f_2(t)=1.54153t^2-1.93684=0$$

的根是虚部近似根，而$x_0=0.5474$已是实部近似值，由$f_2(t)=0$可得

$$t=\pm\sqrt{\frac{1.93684}{1.54153}}\approx\pm 1.12087$$

于是所求最右根的近似值是$0.5474\pm1.12087i$。

运用Ruth表格和$f(z)$在点x_k的展开式，可隔离复根达到相当的精度，将上述方法编成程序可得到复根相当精确的初始近似，再用收敛快的精确化方法就可求得方程（5.9.1）的复根。

5.10

5.10 复多项式的圆盘迭代法

对复多项式

5.10.1 $p(z)=\prod_{j=1}^{n}(z-\xi_j)=0$

求根的圆盘迭代法，最早是由Gargantini(伽甘蒂尼)和Henrici（享里奇）等人于1969年提出的，近十多年有不少发展，这类方法具有收敛速度快，能求出全部根，并且能并行计算等优点，因此，求多项式零点的圆盘算法是一类重要方法。

设$z\in C$，$r\in R$，$r\geqslant0$，则有界闭集

5.10.2 $Z=[z;r]=\{z'\in C|\ |z'-z|\leqslant r\}$

称为C中的一个圆盘。z为圆盘Z的中心，记作 $\mathrm{mid}Z=z$，r为圆盘Z的半径，记为$\mathrm{rad}Z=r$。当$r=0$，定义$[z;0]=z$为点圆。设$Z_i=[z_i;r_i]$（$i=1$，2），定义圆盘四则运算如下：

5.10.3 $Z_1\pm Z_2=[z_1\pm z_2;r_1+r_2]$

$Z_1\cdot Z_2=[z_1\cdot z_2;\ |z_1|r_2+|z_2|r_1+r_1r_2]$

$1/Z_2=\dfrac{1}{|z_2|^2-r_2^2}[\bar{z}_2;r_2]$，　$0\notin Z_2$

$Z_1/Z_2=Z_1\cdot1/Z_2$，　$0\notin Z_2$

其中$\bar{z}_2$表示z_2的共轭复数，$|\cdot|$表示复数模。

作为一种特殊情形，假定$p(z)$的$n-1$个零点位置已知，在一定条件下可确定另一零点的位置。

5.10.4 **定理** 设$W_2,\cdots,W_n$为$n-1$个已知圆盘，若方程(5.10.1)的根$\xi_i\in W_i(i=2,\ \cdots n)$，$z_0\notin W_i$（$i=2$，…，$n$），且满足条

件

5.10.5 $\dfrac{p'(z_0)}{p(z_0)}=c_1$

则$p(z)$的另一零点

$$\xi_1\in W_1=z_0-\frac{1}{c_1-Z_1},$$

其中

$$Z_1=\sum_{j=2}^{n}\frac{1}{z_0-W_j}$$

实际上，由方程（5.10.1）可知

$$c_1=\frac{p'(z_0)}{p(z_0)}=\sum_{j=1}^{n}\frac{1}{z_0-\xi_j}$$

因此有

$$\frac{1}{z_0-\xi_1}=c_1-\sum_{j=2}^{n}\frac{1}{z_0-\xi_j}$$

于是，由$\xi_i\in W_i$　$(i=2,\ \cdots,\ n)$有

$$\xi_1=z_0-\frac{1}{c_1-\sum\limits_{j=2}^{n}\dfrac{1}{z_0-\xi_j}}\in z_0-\frac{1}{c_1-\sum\limits_{j=2}^{n}\dfrac{1}{z_0-W_j}}=W_1$$

因为$z_0\notin W_j$　$(j=2,\ \cdots,)$，故Z_1有意义，它仍然是一个圆盘。如果z_0接近ξ_1，则$p(z_0)$将很小，此时$p(z_0)Z_1$是中心接近原点的半径很小的圆盘，在这种情况下

5.10.6 $W_1=z_0-\dfrac{p(z_0)}{p'(z_0)-p(z_0)Z_1}$

与点Newton修正非常接近，用这种思想去建立计算方法，可看成古典Newton法的圆盘变形。如果假定$W_1^{(0)}=[z^{(0)};r^{(0)}]$已给定，且$W_1^{(0)}$只包含$p(z)$的一个零点$\xi_1$，而开圆盘$|z-z_0|<\rho_0,\rho_0>r^{(0)}$不包含$p(z)$其它零点，定义集合$U=\{z||z-z^{(0)}|\geqslant\rho_0\}$，则$U$包含了$p(z)$的所有其它零点$\xi_2,\ \cdots,\ \xi_n$，且有

$\xi_j \in W_j \subseteq U \quad (j=2, \cdots, n)$

于是由圆盘运算性质知

5.10.7 $$Z_1=\sum_{j=2}^{n}\frac{1}{z^{(0)}-W_j}\subseteq\frac{n-1}{z^{(0)}-U}$$

由此，应用定理（5.10.4）就可构造确定$p(z)$的零点ξ_1的圆盘迭代法。取$W_1^{(0)}$为初始圆盘，则

$$z^{(0)}=\mathrm{mid}W_1^{(0)},\quad Z_1^{(0)}=\frac{n-1}{z^{(0)}-U}$$

由此可构造圆盘迭代算法：

5.10.8 $$\begin{cases} z^{(k)}=\mathrm{mid}W_1^{(k)}, \quad Z_1^{(k)}=\dfrac{n-1}{z^{(k)}-U} \\ W_1^{(k+1)}=z^{(k)}-\dfrac{1}{q(z^k)-Z_1^{(k)}},\ q(z^k)=\dfrac{p'(z^{(k)})}{p(z^{(k)})} \end{cases}$$

$$(k=0,1,\cdots)$$

这算法也称带误差界限的修正Newton算法。若$\xi_1\in W_1^{(0)}$，则对一切k均有$\xi_1\in W_1^{(k)}$。使用时要求选择$W_1^{(0)}$包含ξ_1，但又不包含$p(z)$的其它零点，因此，这种算法仍是局部收敛的。

5.10.9 **定理** 对给定的$W_1^{(0)}=[z^{(0)}, r^{(0)}]$，$\xi_1\in W_1^{(0)}$，$W_1^{(0)}$不包含$p(z)$其它零点，若$r^{(0)}$满足

5.10.10 $6r^{(0)}\leqslant\eta^{(0)}$，$\rho_0=(n-1)\eta^{(0)} \quad (n\geqslant 2)$

则由算法（5.10.8）确定的圆盘序列$\{W_1^{(k)}\}$点收敛于ξ_1，而$r^{(k)}=\mathrm{rad}W_1^{(k)}$满足

5.10.11 $$r^{(k+1)}\leqslant\frac{1}{\eta^{(0)}-2r^{(0)}}(r^{(k)})^2 \quad (k=0,1,\cdots)$$

序列$\{r^{(k)}\}$收敛于0，具有平方敛速。

因此，只要包含ξ_1的圆盘半径满足(5.10.10)，则圆盘序列$\{W_1^{(k)}\}$，平方收敛于ξ_1。如果给出n个互不相交的圆盘$W_j^{(0)}(j=1, 2, \cdots, n)$，且$\xi_j\in W_j^{(0)}$，则由定理（5.10.4）

及算法（5.10.8）可构造求$p(z)$全部零点的圆盘迭代法。

设$W_j^{(0)}=[z_j^{(0)}, r_j^{(0)}]$，$\xi_j\in W_j^{(0)}(j=1, \cdots n)$给定，假定已经算出$W_j^{(k)}=[z_j^{(k)}; r_j^{(k)}]$，$\xi_j\in W_j^{(k)}(j=1, \cdots, n)$，则第$k+1$次迭代可构造成

$$\text{5.10.12} \quad \begin{cases} z_j^{(k)}=\text{mid}W_j^{(k)}, \quad Z_j^{(k)}=\sum\limits_{\substack{i=1\\ i\neq j}}^{n}\dfrac{1}{z_j^{(k)}-W_i^{(k)}} \\ W_j^{(k+1)}=z_j^{(k)}-\dfrac{1}{q(z_j^{(k)})-Z_j^{(k)}}, \quad q(z)=\dfrac{p'(z)}{p(z)} \end{cases}$$

$$(j=1,\cdots,n, \quad k=0,1,\cdots)$$

算法(5.10.12)是算法(5.10.8)的推广。如果已知$\xi_j\in W_j^{(0)}$ $(j=1, \cdots, n)$，则由（5.10.12)可同时迭代出$W_j^{(k)}(j=1, \cdots, n)$，且$\xi_j\in W_j^{(k)}(j=1, \cdots, n)$，它具有明显的并行性质。

5.10.13 定理 对给定的互不相交圆盘$W_i^{(0)}$，$\xi_i\in W_i^{(0)}(i=1, \cdots, n)$.定义

$$r^{(k)}=\max_{1\leqslant i\leqslant n} \ \text{rad}W_i^{(k)} \quad (k=0,1,\cdots)$$

如果$r^{(0)}$满足

$$6r^{(0)}\leqslant\eta^{(0)}$$

其中

$$\eta^{(0)}=\frac{\rho_0}{n-1}, \ n\geqslant2$$

$$\rho_0=\max_{i\neq j}\left\{|z| \ \middle| \ z\in z_j^{(0)}-W_j^{(0)}\right\}, \ z_i^{(0)}=\text{mid}W_i^{(0)}$$

则由算法（5.10.12）确定的圆盘序列$\{W_i^{(k)}\}(i=1, \cdots, n)$点收敛于$\xi_i(i=1, \cdots, n)$，且序列$\{r^{(k)}\}$满足

$$\text{5.10.14} \quad r^{(k+1)}\leqslant\frac{1}{\rho_0(\eta^{(0)}-4r^{(0)})}(r^{(k)})^3 \quad (k=0,1,\cdots)$$

表明序列$\{r^{(k)}\}$收敛于 0 具有立方敛速。

算法（5.10.12）虽然具有收敛快、能同时求解所有零点等优点，但定理条件要求苛刻，具有很强的局部性，故实际使用较困难。

利用复 Lagrange 插值公式还可建立不用计算导数的求 $p(z)$ 全部零点的圆盘迭代法。仍然假定已知 n 个圆盘 $W_i^{(0)}=[z_i^{(0)},r_i^{(0)}]$，$\xi_i\in W_i^{(0)}(i=1,\cdots,n)$，则圆盘迭代程序定义为：

5.10.15 $$W_i^{(k+1)}=z_i^{(k)}-\frac{h_i^{(k)}}{1-Z_i^{(k)}},\quad 0\notin 1-Z_i^{(k)}$$

$$(i=1,\cdots,n;\quad k=0,1,\cdots)$$

其中

$$Z_i^{(k)}=\sum_{\substack{j=1\\j\neq i}}^{n}\frac{h_j^{(k)}}{z_j^{(k)}-W_i^{(k)}},\quad h_i^{(k)}=\frac{p(z_i^{(k)})}{Q'(z_i^{(k)})}$$

$$Q'(z_i)=\prod_{\substack{j=1\\j\neq i}}^{n}(z_i-z_j)$$

可以证明算法（5.10.15）在条件

$\rho_0>3(n-1)r^{(0)}$

下具有立方敛速，它比算法（5.10.12）的条件

$\rho_0\geqslant 6(n-1)r^{(0)}$

减弱，其计算公式也较简单。若在（5.10.15）中将圆盘迭代改为中点序列

5.10.16 $$z_i^{(k+1)}=z_i^{(k)}-\frac{h_i^{(k)}}{1-\sum\limits_{\substack{j=1\\j\neq i}}^{n}\frac{h_j^{(k)}}{z_j^{(k)}-z_i^{(k)}}}$$

$$(i=1,\cdots,n;\quad k=0,1,\cdots)$$

当初值 $z_i^{(0)}(i=1,\ \cdots,\ n)$ 充分靠近零点 $\xi_i(i=1,\ \cdots,\ n)$ 时，具有立方敛速。

5.10.17 例 求多项式

$$p(z)=z^7+z^5-10z^4-z^3-z+10$$

的零点．已知零点$\xi_i\in W_i^{(0)}=[z_i^{(0)};0.3]$，其中

$z_1^{(0)}=2.2$

$z_2^{(0)}=1.2+0.1i$

$z_3^{(0)}=-0.8-0.1i$

$z_4^{(0)}=0.1+1.2i$

$z_5^{(0)}=-0.1-0.8i$

$z_6^{(0)}=-1.1+2.2i$

$z_7^{(0)}=-1.1-1.8i$

解　用圆盘迭代程序(5.10.15)，可得第一次与第二次迭代的最大圆盘的半径为$r^{(1)}\approx5.03\times10^{-2}$，$r^{(2)}\approx2.77\times10^{-5}$。第三次迭代得到下列圆盘：

$$W_1^{(3)}=[2.0000000000000000016-1.11\times10^{-17}i;5.09\times10^{-17}]$$

$$W_2^{(3)}=[1.0000000000000000053+1.29\times10^{-17}i;7.15\times10^{-16}]$$

$$W_3^{(3)}=[-1.0000000000000000003-2.06\times10^{-18}i;3.12\times10^{-17}]$$

$$W_4^{(3)}=[3.06\times10^{-18}+0.9999999999999999999i;2.12\times10^{-17}]$$

$$W_5^{(3)}=[1.63\times10^{-15}-0.99999999999999999981i;1.09\times10^{-16}]$$

$$W_6^{(3)}=[-1.0000000000000000000+2.0000000000000000000i;3.61\times10^{-18}]$$

$$W_7^{(3)}=[-1.0000000000000000000-2.0000000000000000000i;7.92\times10^{-18}]$$

这里最大圆盘半径$r^{(3)}=r_2^{(3)}\approx7.15\times10^{-16}$。迭代三次已得到很高精度，这个问题精确的零点是$\xi_1=2$，$\xi_{2,3}=\pm1$，$\xi_{4,5}=\pm i$，$\xi_{6,7}=-1\pm2i$。

5.11 代数方程求根的稳定性问题

有很多多项式，其系数的微小变化将引起根的很大改变，这种根对系数变化的敏感性称为不稳定性（Instability）。若多项式$p(x)$的系数有微小变化，可表示为

5.11.1 $p_\varepsilon(x)=p(x)+\varepsilon q(x)=0$

其中$q(x)\neq 0$是一个多项式。$p_\varepsilon(x)$的零点表为$x_1(\varepsilon)$，…，$x_n(\varepsilon)$，令$x_1(0)$，…，$x_n(0)$为$p(x)$的零点，即$x_i=x_i(0)$（$i=1$，…，n）.（5.11.1）对ε求导，可得

$$p'(x)\frac{dx}{d\varepsilon}+q(x)+\varepsilon q'(x)\frac{dx}{d\varepsilon}=0$$

$$\frac{dx}{d\varepsilon}=\frac{-q(x)}{p'(x)+\varepsilon q'(x)}$$

于是，当$\varepsilon=0$时有

5.11.2 $$\frac{dx(0)}{d\varepsilon}=\frac{-q(x(0))}{p'(x(0))}$$

应用$x(\varepsilon)$的Taylor展开，当$|\varepsilon|$充分小时，得

5.11.3 $$x_k(\varepsilon)\approx x_k-\frac{q(x_k)}{p'(x_k)}\varepsilon\quad(k=1,\cdots,n)$$

这表明了系数有微小变化ε时引起根的变化的情况，当$x_i(\varepsilon)-x_i$很大时，就称方程是病态的或不稳定的。

5.11.4 例 多项式

$$\begin{aligned}p(x)&=(x-1)(x-2)\cdots(x-7)\\&=x^7-28x^6+322x^5-1960x^4\\&\quad+6769x^3-13132x^2+13068x-5040\end{aligned}$$

解 $q(x)=x^5$，$\varepsilon=-0.002$

$p(x)$的根$x_k=k(k=1,\cdots,7)$，$p'(x_k)=\prod\limits_{j\neq k}(k-j)$

$q(x_k)=k^6$，由(5.11.3)可得

5.11.5 $$x_k(\varepsilon)\approx k+\frac{(-1)^{k-1}0.002k^6}{(k-1)!(7-k)!}=k+\delta(k)$$

$\delta(k)$的数值如下：

$\delta(1)=2.78E-6$，　$\delta(2)=-1.07E-3$

$\delta(3)=3.04E-2$，　$\delta(4)=-2.28E-1$

$\delta(5)=6.51E-1$，　$\delta(6)=-7.77E-1$

$\delta(7)=3.21E-1$

实际上，方程 $p(x)+\varepsilon x^6=0$ 的根 $x_k(\varepsilon)$分别为

1.0000028，　1.9989382，　3.0331253，　3.8195692

5.4586758±0.54012578i，　7.2330128

这说明当x^6的系数相对误差是$\frac{0.002}{28}=7.1E-5$时，根的误差很大，即系数微小变化引起根的不稳定。它说明对病态条件的多项式求根是困难的。由于用计算机计算时，系数在10→2转换中就可能产生微小误差，从而可能使算出的根不可信。为解决这个问题可采用双字长运算，另外，求根时先求量小的根，逐步算出大根，其效果较好。但是，如用降价方法求根则稳定性更差，此时可把得到的根作为近似值，再用Newton法精确化，从而求得精度较高的根。

6

第6章 解线性代数方程组的直接法

6.1

6.1 引 言

n阶线性代数方程组形如

6.1.1 $$\sum_{j=1}^{n} a_{ij}x_j=b_i \quad (i=1,2,\cdots,n)$$

其中系数a_{ij}和b_i是实数或复数．也可将方程组（6.1.1）写成矩阵形式

6.1.2 $$A\boldsymbol{x}=\boldsymbol{b}$$

其中$A=(a_{ij})_n$是n阶矩阵，$\boldsymbol{b}=(b_1, b_2, \cdots, b_n)^T$，$\boldsymbol{x}=(x_1, x_2, \cdots, x_n)^T$．上标"$T$"表示"转置"．

在实际计算中经常遇到求解线性代数方程组的问题．许多有效的数值方法，例如样条逼近，最小二乘法，解非线性方程组的广义Newton（牛顿）法，求Newton-Cotes（牛顿—柯特斯）求积公式系数的方法之一，求微分方程数值解的有限差分法和有限元法等等，都导致求解某些线性代数方程组．众多的事实表明，依赖于已经找到的解线性方程组的精确和有效的方法，线性代数方程组在"计算方法"中及在应用数学和工程中具有特殊的重要性．

线性代数方程组的解法可分为直接法和迭代法两大类．

直接法的特征是：对于阶数n固定的方程组，如果不考虑舍入误差，使用直接法能在预定的运算次数内求得精确解．对中等规模（阶数$n<100$）的方程组及大型的带型或稀疏方程组，由于直接法所具有的准确性和可靠性，所以经常

选用它们来求解。对较高阶的方程组，特别是某些偏微分方程数值求解过程中出现的方程组，由于直接法计算代价较高，使得迭代法更具竞争力。

最基本的直接法是Gauss（高斯）消去法，重要的直接法全都受到Gauss消去法的启示。

6.2

6.2 Gauss消去法

6.2.1 方法的描述

设$A=(a_{ij})_n$是非奇异矩阵，求解n阶线性代数方程组

6.2.1 $Ax=b$

其具体形式是

6.2.2
$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=a_{1,n+1} \\ a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n=a_{2,n+1} \\ \cdots \\ a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n=a_{n,n+1} \end{cases}$$

Gauss消去法的基本思想是依次利用前面的方程消去后面方程中的未知数，将方程组（6.2.2）变换为等价的三角形方程组，而三角形方程组的求解是十分容易的。

第1次消去：设$a_{11}\neq 0$，利用第1个方程消去后面方程中的x_1，得到

6.2.3
$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=a_{1,n+1} \\ \qquad a_{22}^{(1)}x_2+\cdots+a_{2n}^{(1)}x_n=a_{2,n+1}^{(1)} \\ \qquad \cdots \\ \qquad a_{n2}^{(1)}x_2+\cdots+a_{nn}^{(1)}x_n=a_{n,n+1}^{(1)} \end{cases}$$

其中

$$l_{i1}=\frac{a_{i1}}{a_{11}},\quad a_{ij}^{(1)}=a_{ij}-l_{i1}a_{1j}$$

$$(i=2,\cdots,n,\quad j=2,\cdots,n+1)$$

方程组（6.2.3）简记为

$$A^{(1)}x=b^{(1)}$$

这个方程组与（6.2.2）等价（即有相同的解）。

设已得第 $k-1$ 次消去后的结果：

6.2.4 $A^{(k-1)}x=b^{(k-1)}$

其形式是

6.2.5
$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1k}x_k+\cdots+a_{1n}x_n=a_{1,n+1} \\ \quad a_{22}^{(1)}x_2+\cdots+a_{2k}^{(1)}x_k+\cdots+a_{2n}^{(1)}x_n=a_{2,n+1}^{(1)} \\ \quad\cdots \\ \quad a_{kk}^{(k-1)}x_k+\cdots+a_{kn}^{(k-1)}x_n=a_{k,n+1}^{(k-1)} \\ \quad\cdots \\ \quad a_{nk}^{(k-1)}x_k+\cdots+a_{nn}^{(k-1)}x_n=a_{n,n+1}^{(k+1)} \end{cases}$$

若$a_{kk}^{(k-1)}\neq 0$，则可进行第 k 次消去，即利用第 k个方程消去后面方程中的x_k。后面$n-k$个新方程的系数的计算公式是

6.2.6 $l_{ik}=\dfrac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}}$，　$a_{ij}^{(k)}=a_{ij}^{(k-1)}-l_{ik}a_{kj}^{(k-1)}$

$$(i=k+1,\cdots,n,\qquad j=k+1,\cdots,n+1)$$

消去$n-1$次后得与（6.2.2）等价的三角形方程组

6.2.7
$$\begin{cases} a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n=a_{1,n+1} \\ \quad a_{22}^{(1)}x_2+\cdots+a_{2n}^{(1)}x_n=a_{2,n+1}^{(1)} \\ \quad\cdots \\ \quad a_{nn}^{(n-1)}x_n=a_{n,n+1}^{(n-1)} \end{cases}$$

以上过程归结起来（根据6.2.6）是

6.2.8 对$k=1,2,\cdots,n-1$依次计算

$$l_{ik}=\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}},\qquad a_{ij}^{(k)}=a_{ij}^{(k-1)}-l_{ik}a_{kj}^{(k-1)}$$

$$(i=k+1,\cdots,n,\qquad j=k+1,\cdots,n+1)$$

这是对 k 的递推过程，称为**消去过程**。

因矩阵 A 非奇异，故 $a_{nn}^{(n-1)}\neq 0$. 由（6.2.7）得出向后递推公式

$$
\text{6.2.9}\quad \begin{cases} x_n=\dfrac{a_{n,n+1}^{(n-1)}}{a_{nn}^{(n-1)}} \\ x_i=\dfrac{1}{a_{ii}^{(i-1)}}\left\{a_{i,n+1}^{(i-1)}-\sum\limits_{j=i+1}^{n}a_{ij}^{(i-1)}x_j\right\} \end{cases}
$$

$$(i=n-1,n-2,\cdots,1)$$

称为**回代过程**。

先由消去过程（6.2.8）算出方程组（6.2.7）的全部系数，再利用回代过程（6.2.9）算出原方程组（6.2.2）的解，两者一起构成**Gauss消去法**(Gaussian Elimination). 条件是

$$\text{6.2.10}\quad a_{11}a_{22}^{(1)}a_{33}^{(2)}\cdots a_{nn}^{(n-1)}\neq 0$$

由于矩阵A非奇异，所以，必要时通过A中行的适当对换，即方程组(6.2.2)中方程的对换，总可保证条件(6.2.10)成立。但是，通常说的Gauss消去法不包含行的对换. 记

$$
\text{6.2.11}\quad \Delta_i=\begin{vmatrix} a_{11} & \cdots & a_{1i} \\ \cdots & \cdots & \cdots \\ a_{i1} & \cdots & a_{ii} \end{vmatrix}\qquad (i=1,2,\cdots,n)
$$

它们是矩阵A的顺序主子式. 因为Gauss消去法只包含矩阵的行的初等变换，它不改变行列式的值，所以

$$
\Delta_i=\begin{vmatrix} a_{11} & a_{12} & \cdots & a_{1i} \\ & a_{22}^{(1)} & \cdots & a_{2i}^{(1)} \\ & & \cdots & \cdots \\ & & & a_{ii}^{(i-1)} \end{vmatrix}=a_{11}a_{22}^{(1)}\cdots a_{ii}^{(i-1)}
$$

$$(i=1,2,\cdots,n)$$

这样，条件（6.2.10）等价于

6.2.12 $\Delta_i \neq 0 \quad (i=1,2,\cdots,n)$

6.2.13 **例** 用 Gauss 消去法解方程组

$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ x_1 + 3x_2 + 2x_3 = 6 \\ x_1 + 2x_2 + 2x_3 = 5 \end{cases}$$

解 经消去过程，得

$$\begin{cases} 2x_1 + x_2 + x_3 = 4 \\ \dfrac{5}{2}x_2 + \dfrac{3}{2}x_3 = 4 \\ \dfrac{3}{5}x_3 = \dfrac{3}{5} \end{cases}$$

由此通过回代，得到方程组的解

$x_1 = x_2 = x_3 = 1$

利用Gauss消去法解方程组共需乘除法次数为

$$\frac{n^3}{3} + n^2 - \frac{n}{3} \approx \frac{n^3}{3}$$

加减法次数为

$$\frac{1}{6}n(n-1)(2n+5) \approx \frac{n^3}{3}$$

6.2.2 矩阵的三角分解

令

6.2.14 $$L_k = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -l_{k+1,k} & 1 & & \\ & & \vdots & & \ddots & \\ & & -l_{nk} & & & 1 \end{pmatrix}$$

$$(k=1,2,\cdots,n-1)$$

其中l_{ik}的定义在(6.2.8)中．容易验证，第k次消去等价于

$A^{(k)}=L_kA^{(k-1)}$, $\quad b^{(k)}=L_kb^{(k-1)}$

整个消去过程得出的方程组（6.2.7）的系数矩阵$A^{(n-1)}$是一个上三角阵，将它简记作U，显然有

$L_{n-1}\cdots L_2L_1A=U$

由此可得

6.2.15 $A=LU$

其中

6.2.16 $$L=L_1^{-1}L_2^{-1}\cdots L_{n-1}^{-1}=\begin{pmatrix} 1 & & & & \\ l_{21} & 1 & & & \\ l_{31} & l_{32} & 1 & & \\ \vdots & \vdots & \ddots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{n,n-1} & 1 \end{pmatrix}$$

是下三角阵，其对角元素全为1，称为单位下三角阵．

Gauss消去法实质上是将矩阵A分解为一个单位下三角阵L与一个上三角阵U的乘积．当A的所有顺序主子式$\Delta_i\neq0$（$i=1, 2, \cdots, n$），这种分解存在而且唯一．这个事实给人们一种启示，产生通过矩阵直接的三角分解去求解线性代数方程组的新思想．

6.2.3 行列式与逆矩阵计算

设$\Delta_i\neq0(i=1, 2, \cdots, n-1)$，则有分解式(6.2.15)，由此可得到矩阵$A$的行列式

6.2.17 $$\det A=\det L\cdot\det U=\det A^{(n-1)}=a_{11}a_{22}^{(1)}a_{33}^{(2)}\cdots a_{nn}^{(n-1)}$$

亦即，只要消去过程进行完毕，就能算出A的行列式之值．

6.2.18 例 在例(6.2.13)中，方程组系数矩阵的行列式等于

$$2\times\frac{5}{2}\times\frac{3}{5}=3.$$

用 Gauss 消去法计算逆矩阵．设矩阵A非奇异，且条件(6.2.12) 成立．记A的逆矩阵

$A^{-1}=(\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \cdots,\ \boldsymbol{x}_n)$

其中$\boldsymbol{x}_1$，$\boldsymbol{x}_2$，$\cdots$，$\boldsymbol{x}_n$是列向量．根据逆矩阵的定义，有

6.2.19 $A\cdot(\boldsymbol{x}_1,\ \boldsymbol{x}_2,\ \cdots,\ \boldsymbol{x}_n)=(\boldsymbol{e}_1,\ \boldsymbol{e}_2,\ \cdots,\ \boldsymbol{e}_n)$

其中$\boldsymbol{e}_i$是第i个分量为 1 其余分量为 0 的列向量．(6.2.19)等价于n个方程组

$A\boldsymbol{x}_1=\boldsymbol{e}_1$，$A\boldsymbol{x}_2=\boldsymbol{e}_2$，$\cdots$，$A\boldsymbol{x}_n=\boldsymbol{e}_n$

因为这些方程组有相同的系数矩阵，所以 Gauss 消去法的消去过程能同时执行，即在方程组 (6.2.1) 右端用n个向量$\boldsymbol{e}_1,\boldsymbol{e}_2,\cdots,\boldsymbol{e}_n$代替单个向量$\boldsymbol{b}$，实际上只要在 (6.2.8) 中将$j$从$k+1$执行到$n+1$改为执行到$2n$即可．但是回代过程必须分别进行，共执行$n$遍．

6.2.20 **例** 计算例 (6.2.13) 中方程组系数矩阵的逆矩阵．

解 这时解三个方程组，消去过程同时执行，列表如下：

2	1	1	1	0	0
1	3	2	0	1	0
1	2	2	0	0	1
2	1	1	1	0	0
0	5/2	3/2	−1/2	1	0
0	3/2	3/2	−1/2	0	1
2	1	1	1	0	0
0	5/2	3/2	−1/2	1	0
0	0	3/5	−1/5	−3/5	1

利用表中最后三行，依次对右端三列进行回代，得出所求的逆矩阵是

$$\begin{pmatrix} 2/3 & 0 & -1/3 \\ 0 & 1 & -1 \\ -1/3 & -1 & 5/3 \end{pmatrix}$$

6.3

6.3 主元素Gauss消去法

Gauss消去法第 k 步用方程

$$a_{kk}^{(k-1)}x_k + a_{k,k+1}^{(k-1)}x_{k+1} + \cdots + a_{kn}^{(k-1)}x_n = a_{k,n+1}^{(k-1)}$$

消去第$k+1$，…，n个方程中的 x_k，条件是 $a_{kk}^{(k-1)} \neq 0$。$a_{kk}^{(k-1)}$ 称为第 k 步的**主元（素）**(Principal Element 或 Pivot Element)。

Gauss消去法是按方程排列的顺序自然地产生主元素。这样，只要出现 $a_{kk}^{(k-1)}=0$，消去过程则中断；即便 $a_{kk}^{(k-1)} \neq 0$，如果它的绝对值很小，也会因用它作除数，引起其它元素的数量级及舍入误差的急剧增长，而使最后的计算解不可靠，第1章中的例（1.4.4）就是一个具体例子。

主元素法是对 Gauss 消去法的改进。它全面或局部地选取绝对值大的元素为主元素，对 Gauss 消去法的过程作某些技术性修改，使之成为一种有效的算法。在不考虑舍入误差时，主元素法能判断线性代数方程组解的存在性；当解存在时，能求出方程组的解。

6.3.1 全主元素消去法

用主元素法可求解方程组

6.3.1 $\quad Ax=b$

及计算行列式$\det A$与逆矩阵A^{-1}。

设 $\quad A=(a_{ij})_n$

$$b=(a_{1,n+1},\ a_{2,n+1},\ \cdots a_{n,n+1})^T$$

方程组（6.3.1）的增广矩阵为

$$
B=\begin{pmatrix} a_{11} & a_{12} & \cdots & a_{1n} & a_{1,n+1} \\ a_{21} & a_{22} & \cdots & a_{2n} & a_{2,n+1} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nn} & a_{n,n+1} \end{pmatrix} \tag{6.3.2}
$$

Gauss消去法的消去过程是从矩阵 B出发，逐次只进行矩阵中的一行加上另一行的某个常数倍的初等变换，而不进行矩阵中行交换与列交换的初等变换。

在全主元消去法的消去过程中，每一次消去增加了全面选主元的步骤，在作了行交换和列交换后，执行与Gauss 消去法形式相同的计算程序。即，假设经过$k-1$次选主元，行交换，列交换及消去后，将矩阵B约化为

$$
B^{(k-1)}=\begin{pmatrix} a_{11}^{(k-1)} & \cdots & a_{1,k-1}^{(k-1)} & a_{1k}^{(k-1)} & \cdots & a_{1n}^{(k-1)} & a_{1,n+1}^{(k-1)} \\ & \cdots & \cdots & \cdots & \cdots & \cdots & \cdots \\ & & a_{k-1,k-1}^{(k-1)} & a_{k-1,k}^{(k-1)} & \cdots & a_{k-1,n}^{(k-1)} & a_{k-1,n+1}^{(k-1)} \\ & & & \boxed{a_{kk}^{(k-1)}} & \cdots & a_{kn}^{(k-1)} & a_{k,n+1}^{(k-1)} \\ & & & \cdots & \cdots & \cdots & \cdots \\ & & & a_{nk}^{(k-1)} & \cdots & a_{nn}^{(k-1)} & a_{n,n+1}^{(k-1)} \end{pmatrix} \tag{6.3.3}
$$

第 k 次消去步骤是：

首先，全面选主元。在矩阵$B^{(k-1)}$中的方框内选 绝 对值最大的元素$a_{i_kj_k}^{(k-1)}$作主元，即

$$
|a_{i_kj_k}^{(k-1)}|=\max_{\substack{k\leqslant i\leqslant n\\ k\leqslant j\leqslant n}}|a_{ij}^{(k-1)}| \tag{6.3.4}
$$

其次，交换$B^{(k-1)}$的第 k 行与第i_k行，第 k 列与第j_k列，把元素$a_{i_kj_k}^{(k-1)}$放 在 主 元 的位置上。这相当于在方 程组中，交换第k与第i_k二个方程，及第k与第j_k二个未知数。然后，对行、列交换后的矩阵，利用第k行作初等变换将第k列中后$n-k$个元素约化为零。

经$n-1$次消去，最后把方程组（6.3.1）化为三 角 形方

程组

$$
\begin{pmatrix} a_{11}^{(n-1)} & a_{12}^{(n-1)} & \cdots & a_{1n}^{(n-1)} \\ & a_{22}^{(n-1)} & \cdots & a_{2n}^{(n-1)} \\ & & \cdots & \cdots \\ & & & a_{nn}^{(n-1)} \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ \vdots \\ y_n \end{pmatrix} = \begin{pmatrix} a_{1,n+1}^{(n-1)} \\ a_{2,n+1}^{(n-1)} \\ \vdots \\ a_{n,n+1}^{(n-1)} \end{pmatrix} \tag{6.3.5}
$$

其中y_1，y_2，…，y_n是未知数x_1，x_2，…，x_n的某一重新排列。每次列交换要相应调换未知数的位置。

从（6.3.5）回代可得出所求的解。如果出现某个主元为零，则A是奇异矩阵，方程组（6.3.1）无解。

下面给出的适合计算机使用的全主元消去法的算法，是依照上述的方法、步骤，带有行、列交换.实际上，也可设计不带行、列交换的算法。在算法的描述中，记号a_{ij}应理解为存储单元，它最初存放着"数a_{ij}"，在算法执行过程中它存放的内容可以改变，因此当a_{ij}出现在表达式中时，应理解为单元a_{ij}中现行存放着的数。记号"$a:=b$"表示将单元b的内容或数b存入单元a；记号"$a\longleftrightarrow b$"表示交换单元a与单元b的内容。

6.3.6 算法 求解$A\boldsymbol{x}=\boldsymbol{b}$。消元结果冲掉$A$：单位下三角阵$L$中的元素$l_{ij}$冲掉$a_{ij}$，$j<i, i=2,3,\cdots n$；上三角阵$U$即（6.3.5）中的系数矩阵冲掉$a_{ij}$，$i\leqslant j, j=1,2,\cdots,n$。计算解$\boldsymbol{x}$冲掉$\boldsymbol{b}$即$a_{1,n+1}$，$a_{2,n+1}$，…，$a_{n,n+1}$。

消去过程：对于$k=1,2,\cdots,n-1$，

1° 选主元素$a_{i_k j_k}$

$$|a_{i_k j_k}|=\max_{\substack{k\leqslant i\leqslant n\\ k\leqslant j\leqslant n}}|a_{ij}|$$

2° 若$a_{i_k j_k}=0$则计算停止，$\det A=0$。

3° 若$i_k=k$则转4°，否则换行：

$$a_{kj}\longleftrightarrow a_{i_k j} \quad (j=k,k+1,\cdots,n+1)$$

4° 若$j_k=k$则转5°；否则换列：

$a_{ik} \longleftrightarrow a_{ij_k} \quad (i=1,2,\cdots,n)$

这时未知数的次序作了调换

$z[k] \longleftrightarrow z[j_k]$

用记号$z[1],z[2],\cdots,z[n]$表示存放整型数组$z[1:n]$的n个单元，以记录未知数$x_1,x_2,\cdots,x_n$在各种调换过程中的排列次序。开始时，依次存放整数$1,2,\cdots,n$；当交换j_k列与k列时，相应交换$z[j_k]$与$z[k]$中的内容。因此，每次消去后，未知数排列成

$x_{z_1},\ x_{z_2},\ \cdots,\ x_{z_n}$

其中下标z_i是单元$z[i]$中的整数。

5° 计算

$a_{ik}:=l_{ik}=a_{ik}/a_{kk} \qquad (i=k+1,\cdots,n)$

6° 计算

$a_{ij}:=a_{ij}-l_{ik}a_{kj} \qquad (i=k+1,\cdots,n, \qquad j=k+1,\cdots,n+1)$

回代过程：若$a_{nn}=0$则计算停止，否则

1° 计算

$$a_{i,n+1}:=\frac{1}{a_{ii}}\left\{a_{i,n+1}-\sum_{j=i+1}^{n}a_{ij}a_{j,n+1}\right\}$$

$$(i=n,n-1,\cdots,1)$$

2° 根据数组$z[1:n]$的记录，有

$a_{i,n+1}=x_{z_i} \qquad (i=1,2,\cdots,n)$

调整次序

$a_{i,n+1} \longleftrightarrow a_{z_i,n+1}, \quad z[i] \longleftrightarrow z[z_i] \quad (i=1,2,\cdots,n)$

于是计算解$x_1,x_2,\cdots,x_n$依次排列在$a_{1,n+1},a_{2,n+1},\cdots,a_{n,n+1}$中。

6.3.2 列主元素消去法

使用全主元消去法选主元素要用相当多的机器时间。另

一种常用的方法是局部地按列选主元，只须进行方程之间即行之间的交换，因此不产生未知数次序的调换。当第$k-1$次消去得到形如（6.3.3）的矩阵$B^{(k-1)}$之后，第k次消去按列选主元是指在$B^{(k-1)}$的第k列选$a_{k,k}^{(k-1)}, a_{k+1,k}^{(k-1)}, \cdots, a_{nk}^{(k-1)}$中绝对值最大的元素$a_{i_k k}^{(k-1)}$作主元，交换$B^{(k-1)}$中的$k$行与$i_k$行后便可进行消去。

6.3.7 算法 结果存放同算法(6.3.6)，增加一个单元det存放$\det A$的值，

$\det := 1$

消去过程：对于$k=1, 2, \cdots, n-1$,

1° 选主元素$a_{i_k k}$,

$$|a_{i_k k}| = \max_{k \leqslant i \leqslant n} |a_{ik}|$$

2° 若$a_{i_k k}=0$则$\det := 0$，计算停止。

3° 若$i_k = k$则转4°，否则换行：

$$a_{kj} \longleftrightarrow a_{i_k j} \quad (j=k, k+1, \cdots, n+1)$$

且 $\det := -\det$

4° 计算

$$a_{ik} := l_{ik} = a_{ik}/a_{kk} \quad (i=k+1, \cdots, n)$$

5° 计算

$$a_{ij} := a_{ij} - l_{ik}a_{kj} \quad (i=k+1, \cdots, n, \quad j=k+1, \cdots, n+1)$$

6° 计算

$$\det := a_{kk}\det$$

回代过程：计算

$$\det := a_{nn}\det$$

若$a_{nn}=0$则计算停止，否则计算

$$a_{i,n+1} := \frac{1}{a_{ii}}\left\{a_{i,n+1} - \sum_{j=i+1}^{n} a_{ij}a_{j,n+1}\right\}$$

$$(i=n, n-1, \cdots, 1)$$

6.4 Gauss-Jordan消去法

6.4.1 方法的描述

Gauss消去法仅消去对角线下方的元素，Gauss-Jordan（高斯一约当）消去法则同时消去对角线下方及上方的元素，是对Gauss消去法的一个修正。

设方程组$Ax=b$的增广矩阵$B=[A|b]$仍形如(6.3.2)，且A是非奇异矩阵。用Gauss-Jordan消去法求解，完成消去过程时，矩阵B约化为

6.4.1 $$B^{(n-1)}=\begin{pmatrix} a_{11}^{(n-1)} & & & & & a_{1,n+1}^{(n-1)} \\ & a_{22}^{(n-1)} & & & & a_{2,n+1}^{(n-1)} \\ & & a_{33}^{(n-1)} & & & a_{3,n+1}^{(n-1)} \\ & & & \ddots & & \cdots \\ & & & & a_{nn}^{(n-1)} & a_{n,n+1}^{(n-1)} \end{pmatrix}$$

用k表示消去次数，由下面关于k的递推公式可算出矩阵$B^{(n-1)}$中的元素。

6.4.2 对$k=1, 2, \cdots, n-1$依次计算

$$l_{ik}=\frac{a_{ik}^{(k-1)}}{a_{kk}^{(k-1)}} \quad (i=1,2\cdots,n)$$

$$a_{ij}^{(k)}=\begin{cases} a_{ij}^{(k-1)}-l_{ik}a_{kj}^{(k-1)} & (i=1,2,\cdots,n\text{且}i\neq k, j=k,\cdots,n+1) \\ a_{ij}^{(k-1)} & (i=k \text{ 或 } j=1,2,\cdots,k-1) \end{cases}$$

最后，方程组的解为

6.4.3 $$x_i=\frac{a_{i,n+1}^{(n-1)}}{a_{ii}^{(n-1)}} \quad (i=1,2,\cdots,n)$$

为了保证上述求解过程不中断，必须满足条件

$$a_{11}^{(n-1)}a_{22}^{(n-1)}\cdots a_{nn}^{(n-1)}\neq 0$$

如果增加按列选主元的步骤，则只要求矩阵A是非奇异的。

6.4.2 列主元Gauss-Jordan消去法

6.4.4 **算法** 对于$k=1, 2, \cdots, n$

1° 选主元素$a_{i_k k}$,

$$|a_{i_k k}| = \max_{k \leqslant i \leqslant n} |a_{ik}|$$

2° 若$a_{i_k k}=0$则计算停止。

3° 若$i_k=k$则转4°；否则换行

$$a_{kj} \longleftrightarrow a_{i_k j} \quad (j=k, k+1, \cdots, n+1)$$

4° 计算主行（第k行）元素

$$a_{kj} := a_{kj}/a_{kk} \quad (j=k, k+1, \cdots, n+1)$$

5° 消元计算

$$a_{ij} := a_{ij} - a_{ik}a_{kj}$$

$$(i=1,2,\cdots, n \text{且} i \neq k, \ j=k+1, \cdots, n+1)$$

$$a_{ik} := 0 \ (i=1, 2, \cdots, n, \ i \neq k)$$

以上过程结束时矩阵$B=[A \mid b]$约化为

$$\begin{pmatrix} 1 & & & & a_{1,n+1}^{(n-1)} \\ & 1 & & & a_{2,n+1}^{(n-1)} \\ & & \ddots & & \vdots \\ & & & 1 & a_{n,n+1}^{(n-1)} \end{pmatrix}$$

A约化为单位矩阵，方程组的解

$$x_i = a_{i,n+1}^{(n-1)} \quad (i=1,2,\cdots,n)$$

用不着回代。

用Gauss-Jordan消去法，需要$\frac{n^3}{2}+n^2-\frac{n}{2}$次乘除法及同样次数加减法，因此，效果不如Gauss消去法。但用Gauss-Jordan法来求逆矩阵却比较方便。

6.4.3 Gauss-Jordan法求逆矩阵

设A为n阶非奇异矩阵，I为n阶单位矩阵。A的增广矩阵$B=[A \mid I]$，容易证明，若利用列主元Gauss-Jordan消去法将B约化成$[I \mid T]$的形式，则$A^{-1}=T$。

6.4.5 **例** 设

$$A=\begin{pmatrix}1&2&3\\2&4&5\\3&5&6\end{pmatrix}$$

用列主元Gauss-Jordan法求逆矩阵A^{-1}。

解 用记号"$r_i\longleftrightarrow r_j$"表示交换矩阵中的第$i$行与第$j$行．求解过程如下：

$$[A\,|\,I]=\left(\begin{array}{ccc|ccc}1&2&3&1&0&0\\2&4&5&0&1&0\\3&5&6&0&0&1\end{array}\right)\xrightarrow{r_1\leftrightarrow r_3}\left(\begin{array}{ccc|ccc}\boxed{3}&5&6&0&0&1\\2&4&5&0&1&0\\1&2&3&1&0&0\end{array}\right)$$

$$\underline{\text{第一次消去}}\to\left(\begin{array}{ccc|ccc}1&5/3&2&0&0&1/3\\0&\boxed{2/3}&1&0&1&-2/3\\0&1/3&1&1&0&-1/3\end{array}\right)$$

$$\underline{\text{第二次消去}}\to\left(\begin{array}{ccc|ccc}1&0&-1/2&0&-5/2&2\\0&1&3/2&0&3/2&-1\\0&0&\boxed{1/2}&1&-1/2&0\end{array}\right)$$

$$\underline{\text{第三次消去}}\to\left(\begin{array}{ccc|ccc}1&0&0&1&-3&2\\0&1&0&-3&3&-1\\0&0&1&2&-1&0\end{array}\right)=[I\,|\,A^{-1}]$$

其中小方框内是每次按列所选的主元素．

为了节省内存单元，单位矩阵I不必存放．就上述例子来说，第一次消去后将第6列存放在第1列；第二次消去后将第5列存放在第2列；第三次消去后将第4列存放在第3列；最后调整一下列就可在A的位置上得出A^{-1}．

为了最终用A的单元来存放A^{-1}，在第k次消去时，要由这时A的第k列（a_{kk}是主元）

$(a_{1k},\cdots,a_{kk},\cdots,a_{nk})^T$

计算

$$\left(-\frac{a_{1k}}{a_{kk}},\cdots,-\frac{a_{k-1,k}}{a_{kk}},\ \frac{1}{a_{kk}},-\frac{a_{k+1,k}}{a_{kk}},\cdots,-\frac{a_{nk}}{a_{kk}}\right)^T$$

且冲掉A的第k列。

(6.4.6)是列主元 Gauss-Jordan 法求逆矩阵的具体算法.

6.4.6 **算法** 求矩阵A的逆矩阵A^{-1}。计算结果在存放原矩阵A的单元中。用数组$I_n[1:n]$记录主行,A的行列式之值存放在单元det,

det:= 1

消去过程:对于$k=1, 2, \cdots, n$,

1° 选主元素$a_{i_k k}$

$$|a_{i_k k}| = \max_{k\leqslant i\leqslant n} |a_{ik}|$$

$c:= a_{i_k k}$, $I_n[k] := i_k$

2° 若$c=0$则计算停止,$\det A=0$。

3° 若$i_k=k$则转4°;否则转行:

$a_{kj}\longleftrightarrow a_{i_k j} \quad (j=1,2,\cdots,n)$

且

det:= −det

4° 计算

det:= $c\cdot$det

5° 计算主行元素

$a_{kk}:= 1$

$a_{kj}:= a_{kj}/c \quad (j=1, 2, \cdots, n)$

6° 约化非主行元素,对于$i=1, 2, \cdots, n, i\neq k$

$w:= a_{ik}$, $a_{ik} := 0$

$a_{ij} := a_{ij} - a_{kj}w$ ($j=1,2,\cdots,n$)

调整列过程：对于$k=n-1, n-2, \cdots, 1$，

1° $t := I_n[k]$

2° 若$t \neq k$，则换列：

$a_{ik} \leftrightarrow a_{it}$，($i=1, 2, \cdots, n$)

6.5

6.5 直接三角分解法

Gauss消去法求解线性代数方程组

6.5.1 $Ax=\boldsymbol{b}$

时，实际上是将矩阵A分解成为单位下三角阵L与上三角阵U的乘积：

6.5.2 $A=LU$

通常称这种分解为矩阵A的**LU分解**．当A的顺序主子式均不为零时，A的LU分解存在而且唯一．

假设A非奇异，且存在LU分解．那么，只要能够去实现这个分解，便可将求解方程组（6.5.1）的问题归结为极易求解的两个三角形方程组：

6.5.3 $L\boldsymbol{y}=\boldsymbol{b}$，求$\boldsymbol{y}$

和

6.5.4 $U\boldsymbol{x}=\boldsymbol{y}$，求$\boldsymbol{x}$．

直接三角分解法就是建立直接从矩阵A的元素去计算L和U的元素的递推公式，并通过（6.5.3）和(6.5.4)求方程组（6.5.1）的解．

存在针对不同类型矩阵的几种直接三角分解法．当A是一般形式的矩阵时，有不选主元的Doolittle（杜利特尔）分解法，也有选主元的三角分解．如果A是对称正定矩阵，则三角分解可加以简化，有平方根法也称Cholesky（乔莱斯基）分解法及改进的平方根法．对于A是三对角的情形，有

由直接三角分解导出解方程组的追赶法。

6.5.1 Doolittle分解法

设矩阵$A=(a_{ij})_n$有分解式(6.5.2)，并且记

$$L=\begin{pmatrix}1 & & & \\ l_{21} & 1 & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & 1\end{pmatrix},\quad U=\begin{pmatrix}u_{11} & u_{12} & \cdots & u_{1n} \\ & u_{22} & \cdots & u_{2n} \\ & & \ddots & \vdots \\ & & & u_{nn}\end{pmatrix}$$

根据(6.5.2)，利用矩阵乘法可以推出

$$6.5.5 \quad u_{kj}=a_{kj}-\sum_{r=1}^{k-1} l_{kr}u_{rj} \quad (j=k,k+1,\cdots,n)$$

和

$$6.5.6 \quad l_{ik}=\left[a_{ik}-\sum_{r=1}^{k-1} l_{ir}u_{rk}\right]/u_{kk} \quad (i=k+1,k+2,\cdots,n)$$

对于$k=1$，2，$\cdots,n$，交替使用公式(6.5.5)和公式(6.5.6)，即从U的第1行与L的第1列开始，计算共分n步，每步先计算U的一行，再计算L相应的一列，最后可得出L和U的全部元素．这样，便实现了矩阵A的LU分解，其计算过程如下。

u_{11}	u_{12}	u_{13}	$\cdots$	u_{1n}	第1步
l_{21}	u_{22}	u_{23}	$\cdots$	u_{2n}	第2步
l_{31}	l_{32}	$\ddots$			$\vdots$
$\vdots$	$\vdots$		$\ddots$		$\vdots$
l_{n1}	l_{n2}			u_{nn}	第n步

在计算机上，计算好的L和U的元素可存放在A的相应位置。

得出L和U后，由(6.5.3)得计算y的递推公式

6.5.7 $y_i = b_i - \sum_{k=1}^{i-1} l_{ik} y_k \quad (i=1,2,\cdots,n)$

由（6.5.4）得计算x的递推公式

6.5.8 $x_i = \left(y_i - \sum_{k=i+1}^{n} u_{ik} x_k\right) / u_{ii} \quad (i=n, n-1, \cdots, 1)$

所算得的x就是方程组（6.5.1）的解.

6.5.9 **例** 仍考虑例（6.2.13）中的方程组.

解 先求系数矩阵的LU分解，令

$$\begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix} = \begin{pmatrix} 1 & & \\ l_{21} & 1 & \\ l_{31} & l_{32} & 1 \end{pmatrix} \begin{pmatrix} u_{11} & u_{12} & u_{13} \\ & u_{22} & u_{23} \\ & & u_{33} \end{pmatrix}$$

对$k=1$，由（6.5.5）有

$u_{11} = a_{11} = 2, \quad u_{12} = a_{12} = 1, \quad u_{13} = a_{13} = 1$

利用（6.5.6）

$$l_{21} = \frac{a_{21}}{u_{11}} = \frac{1}{2}, \quad l_{31} = \frac{a_{31}}{u_{11}} = \frac{1}{2}$$

对$k=2$，类似地有

$$u_{22} = a_{22} - l_{21} u_{12} = \frac{5}{2}, \quad u_{23} = a_{23} - l_{21} u_{13} = \frac{3}{2}$$

$$l_{32} = \frac{a_{32} - l_{31} u_{12}}{u_{22}} = \frac{3}{5}$$

最后，对$k=3$，有

$$u_{33} = a_{33} - l_{31} u_{13} - l_{32} u_{23} = \frac{3}{5}$$

因此

$$L=\begin{pmatrix}1 & 0 & 0\\ \frac{1}{2} & 1 & 0\\ \frac{1}{2} & \frac{3}{5} & 1\end{pmatrix},\qquad U=\begin{pmatrix}2 & 1 & 1\\ 0 & \frac{5}{2} & \frac{3}{2}\\ 0 & 0 & \frac{3}{5}\end{pmatrix}$$

利用L和U求方程组的解，注意到 $\boldsymbol{b}=(4,6,5)^T$，根据公式（6.5.7）可以算出

$$y_1=4\ ,\qquad y_2=4\ ,\qquad y_3=\frac{3}{5}$$

再应用公式（6.5.8）可得

$x_1=x_2=x_3=1$

上述分解法大约需作$n^3/3$次乘除法，与 Gauss 消去法的计算工作量大体相同.

矩阵A的分解公式（6.5.5）和（6.5.6）即所谓Doolittle分解.

6.5.2 列主元三角分解法

如果公式（6.5.6）中出现等于零或绝对值很小的除数，则Doolittle分解将会中断或引起大的舍入误差.因此，从列主元Gauss消去法出发，建立列主元三角分解法.

因为列主元Gauss消去法仅在消去过程中增加了行交换的步骤，所以用它求解方程组（6.5.1），等价于用Gauss消去法求解先进行一系列的行交换后的方程组

6.5.10 $\quad pAx=pb$

这里 p 是排列矩阵（由单位矩阵进行一系列行交换而得到的矩阵）.于是，有下述关于列主元三角分解的结论：

若 A 为非奇异矩阵，则存在排列矩阵p，使得

6.5.11 $\quad pA=LU$

其中 L 为单位下三角阵，U 为上三角阵。

这样，对于非奇异矩阵 A，可采用选列主元，实现 pA 的 LU 分解，再根据（6.5.10），通过求解 $Ly=pb$ 与 $Ux=y$ 得出原来方程组（6.5.1）的解。

设从 A 出发，选列主元的第 $k-1$ 步分解已完成，并且将已算出的 L 和 U 的元素存放在 A 的相应位置，记作

$$A^{(k-1)}=\left(\begin{array}{cccc|ccc} u_{11} & u_{12} & \cdots & u_{1,k-1} & u_{1k} & \cdots & u_{1n} \\ l_{21} & u_{22} & \cdots & u_{2,k-1} & u_{2k} & \cdots & u_{2n} \\ l_{31} & l_{32} & \ddots & & & & \vdots \\ \vdots & & \ddots & u_{k-1,k-1} & u_{k-1,k} & \cdots & u_{k-1,n} \\ \hline \vdots & & \ddots & l_{k,k-1} & a_{kk} & \cdots & a_{kn} \\ \vdots & & & \vdots & \vdots & & \vdots \\ l_{n1} & l_{n2} & \cdots & l_{n,k-1} & a_{nk} & \cdots & a_{nn} \end{array}\right)$$

此时，由于行交换，$A^{(k-1)}$ 中方框内的 a_{ij} 可能已不是原来 A 中在 (i,j) 位置的元素。

第 k 步分解需利用形如（6.5.5）和（6.5.6）的公式。为了避免用零或绝对值小的数作除数，先计算

6.5.12 $\quad S_i=a_{ik}-\sum_{r=1}^{k-1} l_{ir}u_{rk} \quad (i=k,\ k+1,\cdots,n)$

即（6.5.6）式右端的分子部分及多出一个 S_k. 再在它们中间选取绝对值最大者，记作 S_{i_k}，即

6.5.13 $\quad |S_{i_k}|=\max\limits_{k\leqslant i\leqslant n}|S_i|$

用 S_{i_k} 作为（6.5.6）中的除数 u_{kk}，为此将 $S_k,S_{k+1},\cdots,S_n$ 分别存放到 $A^{(k-1)}$ 中 $a_{kk},\ a_{k+1,k},\ \cdots,\ a_{nk}$ 的位置，然后交换 $A^{(k-1)}$ 的 k 行与 i_k 行的元素，但每个位置上的元素仍用原来的记号。于是

6.5.14 $\quad u_{kk}=a_{kk}=S_{i_k}$

$u_{kj}(j=k+1,\cdots n)$仍可按公式（6.5.5）计算，而l_{ik}的计算可直接利用$S_i(i=k,\ k+1,\ \cdots,n)$

6.5.15 $$l_{ik}=\frac{a_{ik}}{a_{kk}}=\begin{cases}S_i/S_{i_k}, & i=k+1,k+2,\cdots,n,\ i\neq i_k\\ S_k/S_{i_k}, & i=i_k\quad (i_k=k\text{时略去})\end{cases}$$

列主元三角分解法的算法如下。

6.5.16 **算法** 求解方程组$Ax=b$,其中A为非奇异矩阵.pA的LU分解冲掉A，解x冲掉b。

pA的LU分解过程：对于$k=1,2,\cdots n$

1° 计算

$$a_{ik}:=S_i=a_{ik}-\sum_{r=1}^{k-1}l_{ir}u_{rk}\ (i=k,k+1,\cdots,n)$$

2° 选主元素S_{i_k}

$$|S_{i_k}|=\max_{k\leqslant i\leqslant n}|a_{ik}|$$

3° 若$i_k=k$则转4°,否则换行：

$$a_{kj}\longleftrightarrow a_{i_kj}\quad (j=1,2,\cdots,n+1)$$

4° 计算U的第k行元素

$$a_{kk}=u_{kk}=S_{i_k}$$

$$a_{kj}:=u_{kj}=a_{kj}-\sum_{r=1}^{k-1}l_{kr}u_{rj}\quad (j=k+1,\cdots,n)$$

5° 计算L的第k列元素

$$a_{ik}:=l_{ik}=a_{ik}/a_{kk}\quad (i=k+1,\cdots,n)$$

方程组$Ax=b$的求解过程：注意到以上过程已在b的位置上得到pb,

1° 求解$Ly=pb$

$$b_i:=y_i=b_i-\sum_{j=1}^{i-1}l_{ij}b_j\quad (i=1,2,\cdots,n)$$

2° 求解$Ux=y$

$$b_i := x_i = \left(y_i - \sum_{j=i+1}^{n} u_{ij}x_j\right) \Big/ u_{ii} \qquad (i=n, n-1, \cdots, 1)$$

6.5.3 平方根法

设A是n阶对称矩阵，且所有顺序主子式均不为零，则A有LU分解.这时，由$A=A^T$推出$U=DL^T$，因此A可唯一地分解为

6.5.17 $A=LDL^T$

其中L是单位下三角阵，D是对角阵.

假设A是对称正定矩阵，则在A的形如（6.5.17）的分解中，对角矩阵D的对角元素d_1，d_2，…，d_n均为正数.于是可以将D分解为

$$D=D^{\frac{1}{2}}D^{\frac{1}{2}}$$

其中

$$D^{\frac{1}{2}}=\begin{pmatrix} \sqrt{d_1} & & \\ & \ddots & \\ & & \sqrt{d_n} \end{pmatrix}$$

由此及（6.5.17）有

$$A=LDL^T=(LD^{\frac{1}{2}})(LD^{\frac{1}{2}})^T$$

由于$LD^{\frac{1}{2}}$是下三角矩阵，且对角元素均为正数，故可得到下述结论：

若A为对称正定矩阵，则存在唯一的对角元素均为正数的下三角阵L，使

6.5.18 $A=LL^T$

（6.5.18）中的L等于（6.5.17）中的L与$D^{\frac{1}{2}}$的乘积，一般不是单位下三角阵.设

$$L=\begin{pmatrix} l_{11} & & & \\ l_{21} & l_{22} & & \\ \vdots & \vdots & \ddots & \\ l_{n1} & l_{n2} & \cdots & l_{nn} \end{pmatrix}$$

其中$l_{ii}>0$，$i=1,2,\cdots,n$.根据（6.5.18）可以得出计算L元素的递推公式，算出L后，通过求解两个三角形方程组

$\boldsymbol{Ly}=\boldsymbol{b}$，$L^T\boldsymbol{x}=\boldsymbol{y}$

便可得到对称正定方程组$A\boldsymbol{x}=\boldsymbol{b}$的解.具体算法如下.

6.5.19 算法 求解对称正定方程组$A\boldsymbol{x}=\boldsymbol{b}$.由于$A$对称，只须按行存放$A$的对角线及其以下的元素.$L$冲掉$A$，解$\boldsymbol{x}$冲掉$\boldsymbol{b}$.

计算L的过程：对于$j=1$，2，…，n

1° 计算

$$a_{jj}:=l_{jj}=\left(a_{jj}-\sum_{k=1}^{j-1} l_{jk}^2\right)^{\frac{1}{2}}$$

2° 计算

$$a_{ij}:=l_{ij}=\left(a_{ij}-\sum_{k=1}^{j-1} l_{ik}l_{jk}\right)\Big/ l_{jj} \quad (i=j+1,\cdots,n)$$

求解$A\boldsymbol{x}=\boldsymbol{b}$的过程：

1° 计算

$$b_i:=y_i=\left(b_i-\sum_{k=1}^{i-1} l_{ik}y_k\right)\Big/ l_{ii} \quad (i=1,2,\cdots,n)$$

2° 计算

$$b_i:=x_i=\left(y_i-\sum_{k=i+1}^{n} l_{ki}x_k\right)\Big/ l_{ii} \quad (i=n,n-1,\cdots,1)$$

这个算法所描述的直接三角分解法称为**平方根法**或**Cholesky分解法**.

矩阵A的正定性保证$l_{ii}>0$（$i=1$，2，…，n）.平方根法的优点之一是不必选主元，它大约需$n^3/6$次乘除法，还要

进行 n 次需利用逐次逼近的开方运算（见例6.5.21）。

6.5.4 改进的平方根法

为了避免开方运算，对于对称正定矩阵 A 可以采用形如（6.5.17）的分解式，即

$$A=\begin{pmatrix}1 & & & \\ l_{21} & 1 & & \\ \vdots & & \ddots & \\ l_{n1} & \cdots & l_{n,n-1} & 1\end{pmatrix}\begin{pmatrix}d_1 & & & \\ & d_2 & & \\ & & \ddots & \\ & & & d_n\end{pmatrix}\begin{pmatrix}1 & l_{21} & \cdots & l_{n1} \\ & 1 & \ddots & \vdots \\ & & \ddots & l_{n,n-1} \\ & & & 1\end{pmatrix}$$

6.5.20 **算法** 求解对称正定方程组 $A\boldsymbol{x}=\boldsymbol{b}$. 按行存放 A 的对角线及其以下的元素，令

$T=LD=(t_{ij})_n$

算法执行之后，L 的对角线以下的元素及 D 的对角元素存放在 A 的相应位置，解 $\boldsymbol{x}$ 存放在 $\boldsymbol{b}$ 的位置。

计算 L 和 D 的过程：对于 $i=1, 2, \cdots, n$

1° 计算

$$t_{ij}=t_{ij}=a_{ij}-\sum_{k=1}^{j-1} t_{ik}l_{jk} \quad (j=1,2,\cdots,i-1)$$

2° 计算

$$a_{ij}:=l_{ij}=t_{ij}/d_j \quad (j=1,2,\cdots,i-1)$$

3° 计算

$$a_{ii}:=d_i=a_{ii}-\sum_{k=1}^{i-1} t_{ik}l_{ik}$$

求解 $A\boldsymbol{x}=\boldsymbol{b}$ 的过程：

1° 求解 $L\boldsymbol{y}=\boldsymbol{b}$

$$b_i:=y_i=b_i-\sum_{k=1}^{i-1} l_{ik}y_k \quad (i=1,2,\cdots,n)$$

2° 求解 $DL^T\boldsymbol{x}=\boldsymbol{y}$

$$b_i := x_i = \frac{y_i}{d_i} - \sum_{k=i+1}^{n} l_{ki} x_k \quad (i=n, n-1, \cdots, 1)$$

算法（6.5.20）所描述的LDL^T分解法与LL^T分解法相比，所需乘除法运算次数相近，但不需作开方运算，因而称之为**改进的平方根法**。

在实际计算中，求解系数矩阵对称正定的线性代数方程组，多广泛使用平方根法及改进的平方根法。

6.5.21 例 因为例（6.2.13）中的方程组的系数矩阵是对称正定的，所以能应用平方根法和改进的平方根法。系数矩阵为

$$A = \begin{pmatrix} 2 & 1 & 1 \\ 1 & 3 & 2 \\ 1 & 2 & 2 \end{pmatrix}$$

求出 A 的分解式。

解 应用平方根法即算法（6.5.19），得到

$$l_{11} = (a_{11})^{\frac{1}{2}} = \sqrt{2}$$

$$l_{21} = a_{21}/l_{11} = 1/\sqrt{2}$$

$$l_{31} = a_{31}/l_{11} = 1/\sqrt{2}$$

$$l_{22} = (a_{22} - l_{21}^2)^{\frac{1}{2}} = \sqrt{10}/2$$

$$l_{32} = (a_{32} - l_{31}l_{21})/l_{22} = 3/\sqrt{10}$$

$$l_{33} = (a_{33} - l_{31}^2 - l_{32}^2)^{\frac{1}{2}} = \sqrt{15}/5$$

因此，A的LL^T分解为

$$A = \begin{pmatrix} \sqrt{2} & & \\ \frac{1}{\sqrt{2}} & \frac{\sqrt{10}}{2} & \\ \frac{1}{\sqrt{2}} & \frac{3}{\sqrt{10}} & \frac{\sqrt{15}}{5} \end{pmatrix} \begin{pmatrix} \sqrt{2} & \frac{1}{\sqrt{2}} & \frac{1}{\sqrt{2}} \\ & \frac{\sqrt{10}}{2} & \frac{3}{\sqrt{10}} \\ & & \frac{\sqrt{15}}{5} \end{pmatrix}$$

应用改进的平方根法即算法（6.5.20），可得

$d_1=a_{11}=2$

$t_{21}=a_{21}=1,\ \ l_{21}=t_{21}/d_1=1/2$

$d_2=a_{22}-t_{21}l_{21}=5/2$

$t_{31}=a_{31}=1,\ \ l_{31}=t_{31}/d_1=1/2$

$t_{32}=a_{32}-t_{31}l_{21}=3/2,\quad l_{32}=t_{32}/d_2=3/5$

$d_3=a_{33}-t_{31}l_{31}-t_{32}l_{32}=3/5$

因此 A 的LDL^T分解为

$$A=\begin{pmatrix}1 & & \\ \frac{1}{2} & 1 & \\ \frac{1}{2} & \frac{3}{5} & 1\end{pmatrix}\begin{pmatrix}2 & & \\ & \frac{5}{2} & \\ & & \frac{3}{5}\end{pmatrix}\begin{pmatrix}1 & \frac{1}{2} & \frac{1}{2} \\ & 1 & \frac{3}{5} \\ & & 1\end{pmatrix}$$

6.5.5 追赶法

将矩阵的直接三角分解法应用于求解三对角线性代数方程组

$$\begin{pmatrix}b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n\end{pmatrix}\begin{pmatrix}x_1 \\ x_2 \\ \vdots \\ x_n\end{pmatrix}=\begin{pmatrix}d_1 \\ d_2 \\ \vdots \\ d_n\end{pmatrix} \tag{6.5.22}$$

在建立三次样条函数，求微分方程数值解等问题中，都会遇到求解三对角方程组，其系数矩阵常具有对角占优的性质，

$|b_1|>|c_1|,\quad |b_n|>|a_n|$

$|b_i|\geqslant|a_i|+|c_i|>0\quad (i=2,3,\cdots,n)$

这些条件能保证系数矩阵的LU分解存在而且唯一。

将方程组（6.5.22）简写成

$Ax=d$

为了方便，把 A分 解为下三角阵 L 与单位上三角阵 U 的乘积

$$\begin{pmatrix} b_1 & c_1 & & & \\ a_2 & b_2 & c_2 & & \\ & a_3 & \ddots & \ddots & \\ & & \ddots & \ddots & c_{n-1} \\ & & & a_n & b_n \end{pmatrix} = \begin{pmatrix} \mu_1 & & & & \\ \lambda_2 & \mu_2 & & & \\ & \lambda_3 & \ddots & & \\ & & \ddots & \ddots & \\ & & & \lambda_n & \mu_n \end{pmatrix} \begin{pmatrix} 1 & \nu_1 & & & \\ & 1 & \nu_2 & & \\ & & \ddots & \ddots & \\ & & & \ddots & \nu_{n-1} \\ & & & & 1 \end{pmatrix}$$

在这个等式右端利用矩阵乘法，然后与左端比较，便可得出如下确定 L 与U 的元素的公式

6.5.23 $$\begin{cases} \lambda_i = a_i \quad (i=2,3,\cdots,n) \\ \mu_1 = b_1, \quad \mu_i = b_i - a_i\nu_{i-1} \quad (i=2,3,\cdots,n) \\ \nu_1 = c_1/b_1, \quad \nu_i = c_i/(b_i - a_i\nu_{i-1}) \quad (i=2,3,\cdots,n-1) \end{cases}$$

实现 A 的LU分解后，求解方程组（6.5.22）等价于求解两个三角形方程组

$\boldsymbol{Ly}=\boldsymbol{d}$，$\boldsymbol{Ux}=\boldsymbol{y}$

求解方程组（6.5.22）的算法如下。

6.5.24 算法 依次存放数组a_2，a_3，…，a_n；b_1，b_2，…，b_n；c_1，c_2，…，c_{n-1}；d_1，d_2，…，d_n。最终μ_1，μ_2，…，μ_n存放在b_1，b_2，…，b_n的相应位置。ν_1，ν_2，…，ν_{n-1}存放在c_1，c_2，…，c_{n-1}的相应位置，方程组$A\boldsymbol{x}=\boldsymbol{d}$的解 $\boldsymbol{x}$ 存放在 $\boldsymbol{d}$ 的相应位置。a_2，a_3，…，a_n位置的内容不变，亦即λ_2，λ_3，…，λ_n。

1° 对于$i=1$，2，…，$n-1$，计算

$c_i := \nu_i = c_i/\mu_i = c_i/b_i$

$b_{i+1} := \mu_{i+1} = b_{i+1} - a_{i+1}\nu_i$

2° 求解$\boldsymbol{Ly}=\boldsymbol{d}$

$d_1 := y_1 = d_1/\mu_1 = d_1/b_1$

$d_i := y_i = (d_i - a_i y_{i-1})/\mu_i \quad (i=2,3,\cdots,n)$

3° 求解$\boldsymbol{Ux}=\boldsymbol{y}$

$x_n = y_n = d_n$

$d_i := x_i = y_i - \nu_i x_{i+1} \quad (i=n-1,n-2,\cdots,1)$

在以上算法中，计算$\mu_1 \to \nu_1 \to \mu_2 \to \nu_2 \to \mu_3 \to \cdots \to \nu_{n-1} \to \mu_n$及$y_1 \to y_2 \to \cdots \to y_n$的过程称为“追”的过程；计算方程组的解$x_n \to x_{n-1} \to \cdots \to x_1$的过程称为“赶”的过程。所以这种算法称为**追赶法**。

追赶法计算公式特别简单，仅需$5n-4$次乘除法运算，实际上是把Gauss消去法用于求三对角方程组时得出的结果。

6.5.25 例 利用追赶法求解三对角线性代数方程组

$$\begin{pmatrix} 2 & -1 & & \\ -1 & 3 & -2 & \\ & -2 & 4 & -3 \\ & & -3 & 5 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix}=\begin{pmatrix} 6 \\ 1 \\ -2 \\ 1 \end{pmatrix}$$

解 追的过程为

$\mu_1=b_1=2$, $\nu_1=c_1/\mu_1=-1/2$

$\mu_2=b_2-a_2\nu_1=5/2$, $\nu_2=c_2/\mu_2=-4/5$

$\mu_3=b_3-a_3\nu_2=12/5$, $\nu_3=c_3/\mu_3=-5/4$

$\mu_4=b_4-a_4\nu_3=5/4$

$y_1=d_1/\mu_1=3$, $y_2=(d_2-a_2y_1)/\mu_2=8/5$

$y_3=(d_3-a_3y_2)/\mu_3=1/2$, $y_4=(d_4-a_4y_3)/\mu_4=2$

赶的过程为

$x_4=y_4=2$, $x_3=y_3-\nu_3x_4=3$

$x_2=y_2-\nu_2x_3=4$, $x_1=y_1-\nu_1x_2=5$

因此方程组的解

$\boldsymbol{x}=(5,\ 4,\ 3,\ 2)^{\mathrm{T}}$

6.6

6.6 带型方程组的解法

设 n 阶矩阵$A=(a_{ij})_n$.如果 A 满足条件

6.6.1 $a_{ij}=0$，当$|i-j|>m$

其中m为整数，则称A是带宽(Band Width)为$2m+1$的带

型矩阵 (Band Matrix). $m+1$称为A的半带宽.例如，带宽为5($m=2$) 的带型矩阵的一般形式是

$$
6.6.2 \quad A=\begin{pmatrix}
a_{11} & a_{12} & a_{13} & & & & & \\
a_{21} & a_{22} & a_{23} & a_{24} & & & & \\
a_{31} & a_{32} & a_{33} & a_{34} & a_{35} & & & \\
 & a_{42} & a_{43} & a_{44} & a_{45} & a_{46} & & \\
 & & a_{53} & a_{54} & a_{55} & a_{56} & a_{57} & \\
 & & & \cdots & \cdots & \cdots & \cdots & \cdots \\
 & & & & \cdots & \cdots & \cdots & \cdots \\
 & & & & & a_{n,n-2} & a_{n,n-1} & a_{nn}
\end{pmatrix}
$$

具有带型系数矩阵的线性代数方程组简称为**带型方程组**.

一般带型方程组可采用列主元消去法求解，对称正定带型方程组可采用LDL^T分解法求解.对于阶数较高、带宽较窄的方程组，这些求解方法更为有利.

6.6.1 带型方程组的列主元消去法

考虑求解带宽为$2m+1$的一般带型线性代数方程组

6.6.3 $Ax=b$

因为A是一般带型矩阵，所以可采用列主元消去法，以保证解的精度.

算法 (6.3.7) 已经给出一般列主元消去法的描述，它当然也适用于带型方程组.因此，只需要对它作一些技术性的而非实质性的修改，即能利用矩阵A的带型结构，达到节约计算机计算时间及存储单元的目的.

为了节省存储单元，并在消去过程中不在带区外产生非零元素，可在矩阵A的带区前m行和后m行的每行之后增添若干零元素，使A的带区扩充成为n行$2m+1$列的矩形数组$C[1:n,\ 1:2m+1]$.这样，A的第i行元素出现在C中第i

行。数组 C 中的元素c_{ij}如下：

当$1 \leqslant i \leqslant m$ (C 的前m行)

$$c_{ij}=\begin{cases} a_{ij} & (j=1,2,\cdots,m+i) \\ 0 & (j=m+i+1,\cdots,2m+1) \end{cases}$$

当$m+1 \leqslant i \leqslant n-m$ (C 的中间的$n-2m$行)

$$c_{ij}=a_{i,\,i+j} \quad (j=-m,\cdots,-1,0,1,\cdots,m)$$

当$n-m+1 \leqslant i \leqslant n$ (C的后m行)

$$c_{ij}=\begin{cases} a_{i,\,i+j} & (j=-m,\cdots,-1,0,\cdots,n-i) \\ 0 & (j=n-i+1,\cdots,m) \end{cases}$$

在计算机中，数组C的存放占$(2m+1)n$个单元。

以 (6.6.2) 中的矩阵 A 为例.这时，数组$C[1:n,\ 1:5]$及 A 的元素在 C 中的排列方式如下：

i	c_{i1}	c_{i2}	c_{i3}	c_{i4}	c_{i5}
1	a_{11}	a_{12}	a_{13}	0	0
2	a_{21}	a_{22}	a_{23}	a_{24}	0
3	a_{31}	a_{32}	a_{33}	a_{34}	a_{35}
$\cdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$n-2$	$a_{n-2,\,n-4}$	$a_{n-2,\,n-3}$	$a_{n-2,\,n-2}$	$a_{n-2,\,n-1}$	$a_{n-2,\,n}$
$n-1$	$a_{n-1,\,n-3}$	$a_{n-1,\,n-2}$	$a_{n-1,\,n-1}$	$a_{n-1,\,n}$	0
n	$a_{n,\,n-2}$	$a_{n,\,n-1}$	a_{nn}	0	0

具体算法如下，整个消去过程可在数组 C 内进行.

6.6.4 算法 求解 n 阶带宽为$2m+1$的带型方程组 $Ax=b$.矩阵 A 以数组$C[1:n,\ 1:2m+1]$的形式存放. 最终解 x 存放在$b=(b_1,\ b_2,\ \cdots,\ b_n)^T$的相应位置.

消去过程：对于$k=1,\ 2,\ \cdots,\ n-1$

1° 选主元素c_{i_k1}

$$|c_{i_k1}| = \max_{k \leqslant i \leqslant k+m} |c_{i1}|$$

2° 若$i_k = k$则转3°，否则换行

$$c_{kj} \longleftrightarrow c_{i_k j} \quad (j = 1, 2, \cdots, 2m+1)$$

$$b_k \longleftrightarrow b_{i_k}$$

3° 计算

$$c_{kj} := c_{kj}/c_{k1} \quad (j = 2, 3, \cdots, 2m+1)$$

$$b_k := b_k/c_{k1}$$

$$r := \min\{k+m, n\}$$

4° 计算

$$b_i := b_i - c_{i1} b_k \quad (i = k+1, \cdots, r)$$

5° 计算

$$c_{i,j-1} := c_{ij} - c_{i1} c_{kj} \quad (i = k+1, \cdots, r, \quad j = 2, 3, \cdots, 2m+1)$$

6° 冲零

$$c_{i,2m+1} := 0 \quad (i = k+1, \cdots, r)$$

回代过程：计算

$$b_n := x_n = b_n/c_{n1}$$

$$b_i := x_i = b_i - \sum_{j=2}^{2m+1} c_{ij} b_{i+j-1} \quad (i = n-1, n-2, \cdots, 1)$$

6.6.5 例 用算法（6.6.4）求解带型方程组

$$\begin{pmatrix} 7.5 & 3.5 & & \\ 18 & 33 & 4.1 & \\ & 9 & 103 & -1.5 \\ & & 3.7 & 19.3 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 11 \\ 55.1 \\ 110.5 \\ 23 \end{pmatrix}$$

这里$n=4$，$m=1$.数组$C[1:4, 1:3]$的形式及b为

$$\begin{array}{lll|l} 7.5 & 3.5 & 0 & 11 \\ 18 & 33 & 4.1 & 55.1 \\ 9 & 103 & -1.5 & 110.5 \\ 3.7 & 19.3 & 0 & 23 \end{array}$$

第一次消去（$k=1$）在第一行与第二行之间进行，主元素为18，消去结果为

$$
\begin{array}{ccc|c}
18 & 1.8\dot{3} & 0.22\dot{7} & 3.06\dot{1} \\
-10.25 & -1.708\dot{3} & 0 & -11.958\dot{3} \\
9 & 103 & -1.5 & 110.5 \\
3.7 & 19.3 & 0 & 23
\end{array}
$$

第二次消去（$k=2$）在第二行与第三行之间进行，主元素为-10.25，消去结果为

$$
\begin{array}{ccc|c}
18 & 1.8\dot{3} & 0.22\dot{7} & 3.06\dot{1} \\
-10.25 & 0.1\dot{6} & 0 & 1.1\dot{6} \\
101.5 & -1.5 & 0 & 100 \\
3.7 & 19.3 & 0 & 23
\end{array}
$$

第三次消去（$k=3$）在第三行与第四行之间进行，主元素为101.5，最后消去结果为

$$
\begin{array}{ccc|c}
18 & 1.8\dot{3} & 0.22\dot{7} & 3.06\dot{1} \\
-10.25 & 0.1\dot{6} & 0 & 1.0\dot{6} \\
101.5 & -0.014778325 & 0 & 0.985221674 \\
19.3546798 & 0 & 0 & 19.3546798
\end{array}
$$

由回代过程得出方程组的解

$\boldsymbol{x}=(1,\ 1,\ 1,\ 1)^T$

并存放在 $\boldsymbol{b}$ 中。

可以看出，最后在 C 的位置上，第一列存放着各次消去的主元素，以后各列存放着与原方程组等价的系数矩阵为单位上三角阵的方程组的系数。如果出现主元素为零，则 A 是奇异矩阵，计算应停止。

6.6.2 对称正定带型方程组的解法

假设 A 是 n 阶对称正定带形矩阵，带宽为$2m+1$。要解方程组

6.6.6 $Ax=b$

则这时不需要选取主元素。A 可以分解为

6.6.7 $A=LDL^T$

其中 L 是下半带形阵，

$$L=\begin{pmatrix} l_{11} & & & & & \\ l_{21} & l_{22} & & & & \\ \vdots & \ddots & \ddots & & & \\ l_{m+1,1} & & \ddots & \ddots & & \\ & \ddots & & \ddots & \ddots & \\ & & l_{n,n-m} & \cdots & l_{n,n-1} & l_{nn} \end{pmatrix}$$

即当$i<j$或$i-j>m$时，$l_{ij}=0$。D 为对角阵，其对角元素为 $d_{ii}=1/l_{ii}\quad(i=1,2,\cdots,n)$

由（6.6.7）式，可以推出计算 L 的元素的如下公式

6.6.8 $l_{ij}=a_{ij}-\sum_{k=r}^{j-1} l_{ik}l_{jk}/l_{kk}\quad(j=r,\cdots,i;\quad i=1,2,\cdots,n)$

其中

6.6.9 $r=\begin{cases}1, & i\leqslant m+1\\ i-m, & i>m+1\end{cases}$

在分解 A 的同时，对向量 b 作如下分解：

6.6.10 $b=LD\tilde{b}$

容易推出，确定向量 $\tilde{b}$ 的元素的公式为

6.6.11 $\tilde{b}_i=b_i-\sum_{j=r}^{i-1} l_{ij}\tilde{b}_j/l_{jj}\quad(i=1,2,\cdots,n)$

经过上述对 A 和 b 的分解后，方程组（6.6.6）表示成

$LDL^Tx=LD\tilde{b}$

等价于方程组

6.6.12 $L^T x = \widetilde{b}$

由此得出方程组（6.6.6）的解的计算公式

6.6.13 $$x_i = \left(\widetilde{b}_i - \sum_{j=i+1}^{t} l_{ji}\widetilde{b}_j\right) / l_{ii} \quad (i = n, n-1, \cdots, 1)$$

其中

6.6.14 $$t = \begin{cases} n, & i > n-m-1 \\ i+m, & i \leqslant n-m-1 \end{cases}$$

因为 A 对称，所以在计算机中只需存放一个矩形数组 $C[1:n,\ 1:m+1]$，它是通过在 A 的下半带区（对角线及其以下）的左上角添加一些元素而形成的，A 在下半带区第 i 行的元素出现在 C 中的第 i 行.这样,数组 C 中的元素如下：

6.6.15 $$c_{i,\ j-i+m+1} = a_{ij} \quad \begin{matrix} \text{当} i \leqslant m+1 \text{时}, j = 1, 2, \cdots, i \\ \text{当} j > m+1 \text{时}, j = i-m, \cdots, i \end{matrix}$$

由此并根据（6.6.8）、（6.6.11）、（6.6.13），可写出适用于计算机的算法.

6.6.16 **算法** 求解 n 阶带宽为 $2m+1$ 的对称正定带型方程组 $Ax = b$.矩阵 A 的下半带区以数组 $C[1:n,\ 1:m+1]$ 的形式存放.最终 L 存放在 C 的位置，解 x 存放在 b 的位置.

计算 L 的过程：对于 $i = 1,\ 2,\ \cdots n$

1° 计算

$$c_{i,\ j-i+m+1} := l_{ij} = c_{i,\ j-i+m+1} - \sum_{k=r}^{j-1} c_{i,\ k-i+m+1}\ c_{j,\ k-j+m+1} / c_{k,\ m+1}$$

当 $i \leqslant m+1$ 时 $r = 1 \quad (j = 1, 2, \cdots, i)$

当 $i > m+1$ 时 $r = i-m \quad (j = i-m, \cdots, i)$

2° 计算

$$b_i := \widetilde{b_i} = b_i - \sum_{j=r}^{i-1} c_{i,\,j-i+m+1} b_j / c_{j,\,m+1}$$

其中 r 按（6.6.9）式确定。

求解$L^t x = b$的过程：对于$i=n$，$n-1$，…，1

$$b_i := x_i = \left(b_i - \sum_{j=i+1}^{t} c_{j,\,i-j+m+1} b_j \right) \Big/ c_{i,\,m+1}$$

其中 t 按（6.6.14）式确定。

6.6.17 例 用算法（6.6.16）求解方程组

$$\begin{pmatrix} 5 & 6 & & \\ 6 & 5 & 6 & \\ & 6 & 5 & 6 \\ & & 6 & 5 \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \\ x_4 \end{pmatrix} = \begin{pmatrix} 11 \\ 17 \\ 17 \\ 11 \end{pmatrix}$$

解 这时$n=4$,$m=1$,数组$C[1:4,1:2]$的形式及 b 如下：

$$\begin{array}{cc|c} a & 5 & 11 \\ 6 & 5 & 17 \\ 6 & 5 & 17 \\ 6 & 5 & 11 \end{array}$$

其中 a 可取任意值，例如取0或1。

计算 L 的过程：

第一步，$i=1$，$r=1$，$j=1$

$c_{12} := l_{11} = c_{12} = 5$，　$b_1 := \widetilde{b_1} = b_1 = 11$

第二步，$i=2$，$r=1$，$j=1$，2

$c_{21} := l_{21} = c_{21} = 6$，　$c_{22} := l_{22} = c_{22} - c_{21}^2/c_{12} = -2.2$

$b_2 := \widetilde{b_2} = b_2 - c_{21}b_1/c_{12} = 3.8$

第三步，$i=3$，$r=2$，$j=2,3$

$c_{31} := l_{32} = c_{31} = 6$，$c_{32} := l_{33} = c_{32} - c_{31}^2/c_{22} = 21.\dot{3}\dot{6}$

$b_3 := \widetilde{b_3} = b_3 - c_{31}b_2/c_{22} = 27.\dot{3}\dot{6}$

第四步，$i=4$，$r=3$，$j=3$，4

$c_{41} := l_{43} = c_{41} = 6$，$c_{42} := l_{44} = c_{42} - c_{41}^2/c_{32} = 3.31489$

$b_4 := \widetilde{b_4} = b_4 - c_{41}b_3/c_{32} = 3.31489$

因此，C 与 $\boldsymbol{b}$ 位置上的内容变成

α	5	11
6	-2.2	3.8
6	$21.\dot{3}\dot{6}$	$27.\dot{3}\dot{6}$
6	3.31489	3.31489

再利用求解$L^T\boldsymbol{x}=\boldsymbol{b}$的过程，得到

$b_4 := x_4 = b_4/c_{42} = 1$

$b_3 := x_3 = (b_3 - c_{41}b_4)/c_{32} = 1$

$b_2 := x_2 = (b_2 - c_{31}b_3)/c_{22} = 1$

$b_1 := x_1 = (b_1 - c_{21}b_2)/c_{12} = 1$

因此所求方程组的解为

$\boldsymbol{x} = (1,1,1,1)^T$

例（6.6.17）中方程组的系数矩阵对称但非正定，说明算法（6.6.16）不仅适用对称正定带型方程组，也适用于具有对称非奇异带型系数矩阵的方程组。

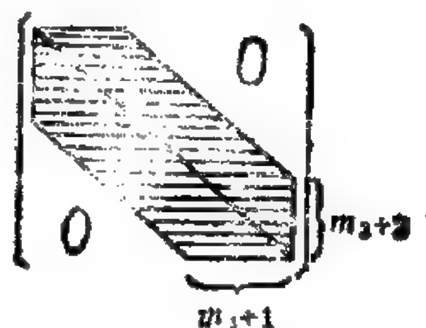
6.7 大型稀疏方程组的解法

6.7.1 稀疏矩阵

大量元素为零、只有为数不多的元素不为零的矩阵，通常被称之为**稀疏矩阵**（Sparse Matrix）。

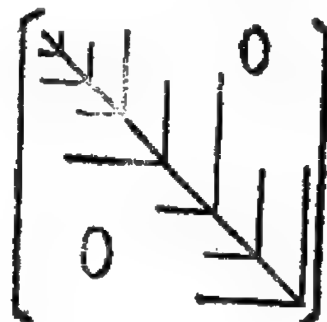
实际问题中遇到大量大型（即阶 n 很大的）稀疏矩阵，非零元素常常仅占5～10%，或与矩阵的阶 n 同数量级（如每行有 2 至10个非零元素等）。在结构分析、电力传输网分析、大地测量、微分方程数值解、图论、遗传理论、社会及行为科学等问题中，都常出现稀疏矩阵，因而受到人们的关注。

区别一个矩阵是否为稀疏矩阵，并不严格取决于非零元素所占的百分比。只要是有很多零元素而且在分布上具有能够被有效利用的特点的矩阵，都可应用稀疏矩阵技术来处理。实际中常见的稀疏矩阵有以下几种形式：



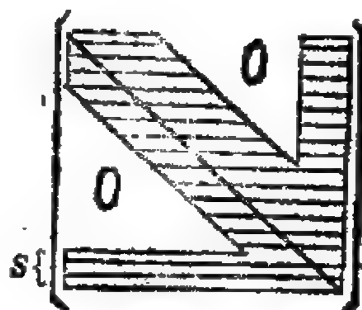
6.7.1 图 一般带型矩阵（非零元素在阴影区域）

其中 m_1+1 是下半带宽，m_2+1 是上半带宽，总带宽是 m_1+m_2+1。



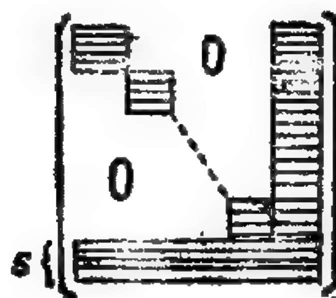
6.7.2 图 变带宽带型矩阵

带区各行各列非零元素"宽度"不等。



6.7.3 图 加边带状矩阵

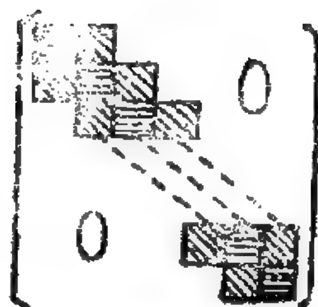
带状矩阵加上一个宽度为s的边。



6.7.4 图 加边对角块矩阵

分块对角矩阵加一个宽度为 s 的边。

6.7.5 图　分块三对角矩阵



更为一般的形式是：某种绝大多数元素为零，但非零元素分布不大规则的矩阵。

具有稀疏系数矩阵的线性代数方程组简称为**稀疏方程组**，常用的解法都是针对稀疏矩阵的具体类型而提出的特殊方法。各种解法仍然依据直接方法的一般原理，但它们从稀疏矩阵零元素很多及分布的特点出发，采用了适当的处理技术，从而大大节省了计算工作量和存储量，有效地提高在计算机上求解问题的阶数。

6.7.2　压缩的存储形式

节省稀疏矩阵在计算机中的存储量，使计算机能存储和处理更大型的问题，是建立稀疏方程组有效解法的关键之一。通常以压缩的形式在计算机中存储大型稀疏矩阵，亦即只存储稀疏矩阵的非零元素及必要的索引信息(例如，为了存储非零元素a_{ij}，需要同时以适当的方式存储行指标i和列指标j)。

有各种有效的压缩方案可供采用，使用逻辑尺的压缩方案是其中一种易于理解的压缩方案，而其它方案则可参见本书最后列出的有关稀疏矩阵的文献。

设A是n阶稀疏矩阵．逻辑尺共n^2个二进制位，是由$[n^2/p]+1$（$[x]$表示x的整数部分）个存储单元组成的，其中p是计算机的字长．逻辑尺的n^2个二进制位的每一位依次对应于A的一个元素，若该元素不为零，则该位为1，否则

为零.同时，仅将 A 的非零元素按与逻辑尺的对应顺序依次存储起来.这样，在求解过程中，逻辑尺可以确定每个非零元素的位置，并且通过对它的分析和修改，判别哪些中间结果需要计算和存储，节省与零元素相应的那些不必要的运算，达到节省计算工作量与存储量的目的.

以消去法为例.在消去过程中，A 的零元素的位置上一般将出现一些新的非零元素.为了使中间过程产生的非零元素尽可能少，需适当地重新排列方程与未知数的次序.比如非零元素少的行（方程）排在前面，消去时将使其它行产生较少新的非零元素等等.排完次序后，便可按通常顺序的消去过程进行计算.在计算过程中将反复使用逻辑尺，使零元素不参加运算，记录新产生的非零元素并修改逻辑尺等等，最终得出上三角矩阵U的元素及相应的逻辑尺.常数项向量b在上述过程中相应改变.再根据新的逻辑尺进行去零回代，完成求解过程.

以上是使用逻辑尺的压缩方案的基本思想，其具体细节相当繁琐.对于非常稀疏的矩阵，使用逻辑尺可能会浪费存储单元.这时可用一系列单元直接存放非零元素的行指标和列指标，其过程与用逻辑尺的办法大致类似.

对于某些非零元素分布不规则的稀疏矩阵，才有必要使用逻辑尺等较繁琐的压缩方案.至于非零元素分布有一定规则的稀疏矩阵，则往往可以采用比较简单的压缩存储方式.这一点在第 6 节关于带型方程组的解法中已见到.

6.7.3 解稀疏方程组的三角分解法

用直接三角分解法求解大型稀疏方程组 $Ax=b$，为了简便，假定可以不必选主元素，而采用LU分解法.

从计算l_{ij}的公式（6.5.6）可以看出：若矩阵 A 第 i 行的前k_i个元素为零，

$a_{ij}=0, \quad 1\leqslant j\leqslant k_i<i$

则矩阵 L 相应地有

$l_{ij}=0, \quad 1\leqslant j\leqslant k_i<i$

从计算u_{ij}的公式（6.5.5），对矩阵 A 与 U 的相应的列，可以得出类似的结论：若

$a_{ij}=0, \quad 1\leqslant i\leqslant k_j<j$

则

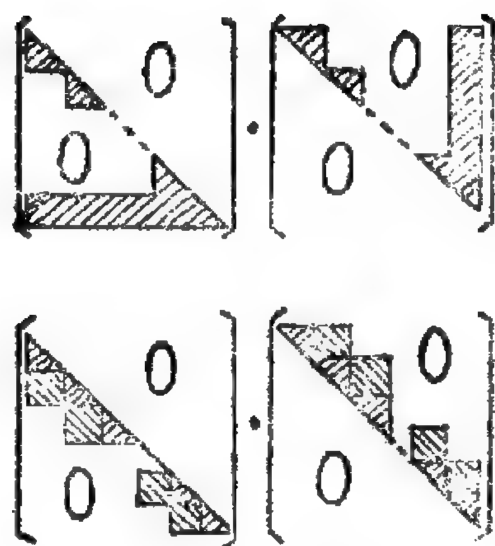
$u_{ij}=0, \quad 1\leqslant i\leqslant k_j<j$

这样，对于形如图（6.7.1）～（6.7.5）中的矩阵 A 来说，下三角矩阵 L 与矩阵 A 的下三角部分应该有相同的形状（指非零元素所在的区域）；上三角矩阵 U 与矩阵 A 的上三角部分应该有相同的形状。因此，所说五种特殊形状矩阵相应的 LU 分解应如图（6.7.6）所示。

6.7.6 图

m_1+1 m_2+1

0 0 0 0 0 0 0 0 0 0 0 0



利用图（6.7.6）给出的LU分解的特点，在对图(6.7.1)～（6.7.5）型矩阵进行分解时仅需计算其中的非零区的元素，从而大大节省了计算工作量与存储量.由于要利用矩阵的特殊形状来进行计算，就必须针对不同的形状构造相应的具体算法.这些算法的构造原则大体相同.

但是，对于非零元素分布不大规则的稀疏矩阵，上述LU分解的结论是不成立的.这是因为在分解过程中，原来矩阵中某些零元素的位置将出现新的非零元素．矩阵L与U的非零元素分布将与A的相应部分有所不同.因此，在这种情况下，必须采用适当的压缩存储方案及适用于一般稀疏矩阵的算法.

6.7.4 对称正定稀疏方程组的解法

对称正定稀疏方程组的解法，适用于求解大型线性代数方程组

$$AX=B \tag{6.7.7}$$

其中A是n阶对称正定稀疏矩阵，X和B是$n\times m(m\geqslant 1)$的矩阵.也就是说，可以同时处理系数矩阵为A的m个方程组.

对A作如下分解：

6.7.8 $A=LDL^T$

其中$L=(l_{ij})_n$为单位下三角阵，D为对角矩阵

$D=\mathrm{diag}(d_1,d_2,\cdots,d_n)$

设矩阵A各行的第一个非零元素是

$a_{im_i}\quad(i=1,2,\cdots,n)$

记

$m_{ij}=\max(m_i,m_j)$

则矩阵L和D的元素由下列公式确定：

6.7.9
$$\begin{cases} l_{ij}=\left(a_{ij}-\sum\limits_{k=m_{ij}}^{j-1} l_{ik}d_k l_{jk}\right)/d_j \\ \qquad (j=m_i,m_i+1,\cdots,i-1,\qquad i=2,3,\cdots,n) \\ d_i=a_{ii}-\sum\limits_{k=m_i}^{i-1} l_{ik}^2 d_k \qquad (i=1,2,\cdots,n) \end{cases}$$

在分解A的同时，对矩阵$B=(b_{ij})_{n\times m}$也作下列分解：

6.7.10 $B=LD\widetilde{B}$

确定矩阵$\widetilde{B}=(\widetilde{b}_{ij})_{n\times m}$的元素的公式为

6.7.11 $$\widetilde{b}_{ik}=\left(b_{ik}-\sum_{j=m_i}^{i-1} l_{ij}d_j\widetilde{b}_{jk}\right)/d_i$$

$$(i=1,2,\cdots,n,\qquad k=1,2,\cdots,m)$$

将（6.7.8）和（6.7.10）代入（6.7.7），得到

6.7.12 $L^TX=\widetilde{B}$

这样，方程组（6.7.7）的求解归结为对方程组（6.7.12）的求解。注意到L^T的第i列的第一个非零元素是$a_{im_i}(i=1,2,\cdots,n)$，为了节省计算量，对方程组（6.7.12）可采用以下求解过程（$\widetilde{b}_{jk}$应理解为开始存放着元素"$\widetilde{b}_{jk}$"的存储单元）：

对于$i=n,n-1,\cdots,1$，依次计算

6.7.13 $\widetilde{b}_{jk}:=\widetilde{b}_{jk}-l_{ij}\widetilde{b}_{ik}$ $\left(\begin{array}{l}j=m_i,m_i+1,\cdots,i-1\\ k=1,2,\cdots,m\end{array}\right)$

最后有

6.7.14 $x_{ik}=\widetilde{b}_{ik}$ $(i=1,2,\cdots,n,\quad k=1,2,\cdots,m)$

为了节省存储量，对 A 采用变带宽的压缩存储方式：将 A 各行的第一个非零元至对角元依次按行排列成一维数组

a_{11}, a_{2m_2}, a_{22}, a_{3m_3}, $\cdots$, a_{33}, $\cdots$, a_{im_i}, $\cdots$, a_{ii}, $\cdots$, a_{nm_n}, $\cdots$, a_{nn}

记作$a[1:p]$，其中$p=\sum\limits_{i=1}^{n}(i-m_i+1)$是数组 a 中元素的个数，A 以数组 a 的形式存放。为了判定数组 a 的元素在 A 中相应位置，需用一个整型数组$d[0:n]$，其中$d[0]=0$，$d[i]$标记对角元a_{ii}在数组 a 中的序号，$i=1,2,\cdots,n$。不难看出，矩阵 A 完全由数组 a 和 d 的内容唯一确定，而且 A 在数组a中出现的元素a_{ij}处在 a 的第$d[i]-i+j$个位置上，即

6.7.15 $a_{ij}=a[d[i]-i+j]$

A 每行第一个非零元素的列指标与数组 d 的关系由下面公式确定：

6.7.16 $m_i=i-(d[i]-d[i-1])+1\quad(i=1,\ 2,\ \cdots,\ n)$

矩阵 B 则直接构成一个数组$b[1:n,\ 1:m]$，并按此存放。

完整的算法如下：

6.7.17 **算法** 求解对称正定稀疏方程组$AX=B$。最终解X 存放在数组$b[1:n,\ 1:m]$的位置。

计算 L 和D 的过程：对于$i=1,\ 2,\ \cdots,\ n$

1° 计算

$I:=d[i]-i$，$MI:=m_i=i-(d[i]-d[i-1])+1$

2° 对于$j=m_i,\ m_i+1,\ \cdots,\ i$

$J:=d[j]-j$, $MJ:=m_j=j-(d[j]-d[j-1])+1$

$M:=m_{ij}=\max(MI, MJ)=\max(m_i, m_j)$

$$a[I+j]:=a[I+j]-\sum_{k=m_{ij}}^{j-1} a[I+k]\cdot a[d[k]]\cdot a[J+k]$$

若$j=i$，则转3°，否则

$a[I+j]:=a[I+j]/a[d[j]]$

$b[i,k]:=b[i,k]-a[I+j]\cdot a[d[j]]\cdot b[j,k]$

$$(k=1,2,\cdots,m)$$

3° 若$a[I+i]=0$，则计算停止(这时矩阵A非正定)，否则

$b[i,k]:=b[i,k]/a[d[i]]$ $(k=1,2,\cdots,m)$

解方程组$L^TX=\widetilde{B}$的过程：对于$i=n, n-1, \cdots, 1$

1° 计算

$I:=d[i]-i$, $MI:=m_i=i-(d[i]-d[i-1])+1$

2° 对于$j=m_i, m_i+1, \cdots, i-1$

$b[j,k]:=b[j,k]-a[I+j]\cdot b[i,k]$ $(k=1, 2, \cdots, m)$

6.7.18 例 解方程组$A\boldsymbol{x}=\boldsymbol{b}$，其中

$$A=\begin{pmatrix} 4.5 & 0.2 & -1.3 & 0 & 0 & 0 \\ 0.2 & 5.3 & 0 & 0 & 0 & 0 \\ -1.3 & 0 & 10.2 & 5.1 & 0 & -1.7 \\ 0 & 0 & 5.1 & 8.4 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0.6 & 0 \\ 0 & 0 & -1.7 & 0 & 0 & 3.1 \end{pmatrix}, \boldsymbol{b}=\begin{pmatrix} 3.4 \\ 5.5 \\ 12.3 \\ 13.5 \\ 0.6 \\ 1.4 \end{pmatrix}$$

解 这时$n=6$，$m=1$。矩阵A对称正定，A中黑体字的元素为该行第一个非零元素，存放A的有关元素的数组$a[1:13]$为

4.5，0.2，5.3，−1.3，0，10.2，5.1，8.4，0.6，−1.7，0，0，3.1

记录A的对角元素在a中的序号的整型数组$d[0:6]$为

0, 1, 3, 6, 8, 9, 13

利用算法（6.7.17）得到的计算结果与精确解一致：

$x=(1,1,1,1,1,1)^T$

6.8

6.8 复线性代数方程组的解法

6.8.1 化为实系数方程组求解

复线性代数方程组的一般形式是

6.8.1 $(A+iB)(x+iy)=c+id$

其中 A 和 B 是 n 阶实矩阵，c 和 d 是 n 维实向量，x 和 y 是未知解向量的实部和虚部。

方程组（6.8.1）可以写成

$(Ax-By)+i(Bx+Ay)=c+id$

因此求解方程组（6.8.1）等价于求解 $2n$ 阶实系数线性代数方程组

6.8.2
$$\begin{pmatrix} A & -B \\ B & A \end{pmatrix}\begin{pmatrix} x \\ y \end{pmatrix}=\begin{pmatrix} c \\ d \end{pmatrix}$$

这样，只要阶数 n 不很高，便可应用线性代数方程组各种已有的解法，通过求解实方程组（6.8.2），得出复方程组（6.8.1）的解。

但是，这种转换求解的方法比起直接去求解方程组（6.8.1），存储量要增加一倍，计算时间也要增加不少。因此，当阶数 n 较高且内存容量较小时，便不易实现方程组（6.8.1）的求解，而需要使用直接求解方程组（6.8.1）的方法。

6.8.2 复线性方程组的列主元消去法

直接求解复方程组（6.8.1）的列主元消去法，是在每列按模最大选取列主元素，再按Gauss消去法进行计算，其步骤与实系数方程组列主元消去法相同，不同之点仅在于每

步运算采用了复数运算.

方程组 (6.8.1) 的复增广矩阵为

$$[A+iB \mid \boldsymbol{c}+i\boldsymbol{d}]=[A \mid \boldsymbol{c}]+i[B \mid \boldsymbol{d}]$$

设$[A \mid \boldsymbol{c}]=(a_{lj})_{n\times(n+1)}$, $[B \mid \boldsymbol{d}]=(b_{lj})_{n\times(n+1)}$,它们分别可以用数组$a[1:n, 1:n+1]$, $b[1:n, 1:n+1]$的形式存放.解向量的实部$\boldsymbol{x}$和虚部$\boldsymbol{y}$, 分别用数组$x[1:n]$和$y[1:n]$的形式存放.

6.8.3 算法 解复线性方程组 (6.8.1) .

消去过程: 对于$k=1, 2, \cdots, n$

1° 选主元$a_{l_k j}+ib_{l_k j}$

$$t:=a^2_{l_k j}+b^2_{l_k j}=\max_{k\leq l\leq n}(a^2_{lj}+b^2_{lj})$$

2° 若$t=0$则计算停止.

3° 若$l_k\neq k$则换行

$$\begin{aligned}&a_{kj}\leftrightarrow a_{l_k j}\\ &\qquad\qquad (j=k,k+1,\cdots,n+1)\\ &b_{kj}\leftrightarrow b_{l_k j}\end{aligned}$$

4° 计算主行(第k行)元素: 对于$j=k+1, \cdots, n+1$

$$r:=(a_{kj}a_{kk}+b_{kj}b_{kk})/t$$

$$b_{kj}:=(b_{kj}a_{kk}-a_{kj}b_{kk})/t$$

$$a_{kj}:=r$$

5° 对于$l=k+1, \cdots, n$

$$\begin{aligned}&a_{lj}:=a_{lj}-a_{lk}a_{kj}+b_{lk}b_{kj}\\ &\qquad\qquad\qquad\qquad (j=k+1,\cdots,n+1)\\ &b_{lj}:=b_{lj}-a_{lk}b_{kj}-b_{lk}a_{kj}\end{aligned}$$

回代过程: 对于$l=n,n-1, \cdots, 1$

$$x_l:=a_{l,n+1}\ ,\ y_l:=b_{l,n+1}$$

$$x_l:=x_l-\sum_{k=l+1}^{n}(a_{lk}x_k-b_{lk}y_k)$$

$$y_l := y_l - \sum_{k=l+1}^{n} (a_{lk} y_k + b_{lk} x_k)$$

6.8.4 例 用算法（6.8.3）解复线性方程组

$$\begin{pmatrix} 1+i & 1+i & 1+i \\ 2+2i & 1+i & 1+i \\ 3+3i & 2+2i & 1+i \end{pmatrix} \begin{pmatrix} x_1+iy_1 \\ x_2+iy_2 \\ x_3+iy_3 \end{pmatrix} = \begin{pmatrix} 12i \\ 14i \\ 20i \end{pmatrix}$$

解 这时增广矩阵为

$$[A \mid c] = \begin{pmatrix} 1 & 1 & 1 & 0 \\ 2 & 1 & 1 & 0 \\ 3 & 2 & 1 & 0 \end{pmatrix}, \quad [B \mid d] = \begin{pmatrix} 1 & 1 & 1 & 12 \\ 2 & 1 & 1 & 14 \\ 3 & 2 & 1 & 20 \end{pmatrix}$$

第一次消去（$k=1$）：主元$3+3i$，$t=18$，交换$[A \mid c]$和$[B \mid d]$中一、三两行.结果为

$$\begin{pmatrix} 3 & \frac{2}{3} & \frac{1}{3} & \frac{10}{3} \\ 1 & \frac{1}{3} & \frac{2}{3} & 0 \\ 2 & -\frac{1}{3} & \frac{1}{3} & 0 \end{pmatrix}, \quad \begin{pmatrix} 3 & 0 & 0 & \frac{10}{3} \\ 1 & \frac{1}{3} & \frac{2}{3} & \frac{16}{3} \\ 2 & -\frac{1}{3} & \frac{1}{3} & \frac{2}{3} \end{pmatrix}$$

这是第一次消去结束时出现在$[A \mid c]$和$[B \mid d]$相应位置上的元素.

第二次消去（$k=2$）：主元$\frac{1}{3}+\frac{1}{3}i$，$t=\frac{2}{9}$.结果为

$$\begin{pmatrix} 3 & \frac{2}{3} & \frac{1}{3} & \frac{10}{3} \\ 1 & \frac{1}{3} & 2 & 8 \\ 2 & -\frac{1}{3} & 1 & 0 \end{pmatrix}, \quad \begin{pmatrix} 3 & 0 & 0 & \frac{10}{3} \\ 1 & \frac{1}{3} & 0 & 8 \\ 2 & -\frac{1}{3} & 1 & 6 \end{pmatrix}$$

第三次消去（$k=3$）：主元$1+i$，$l=2$.结果为

$$\begin{pmatrix} 3 & \frac{2}{3} & \frac{1}{3} & \frac{10}{3} \\ 1 & \frac{1}{3} & 2 & 8 \\ 2 & -\frac{1}{3} & 1 & 3 \end{pmatrix}, \quad \begin{pmatrix} 3 & 0 & 0 & \frac{10}{3} \\ 1 & \frac{1}{3} & 0 & 8 \\ 2 & -\frac{1}{3} & 1 & 3 \end{pmatrix}$$

这是最后出现在$[A|c]$和$[B|d]$相应位置上的元素，回代时利用其中黑体的元素.

利用回代过程得到

$x=(1,\ 2,\ 3)^T$，$y=(1,\ 2,\ 3)^T$

因此方程组的解为

$x+iy=(1+i,\ 2+2i,\ 3+3i)^T$

容易验证这是方程组的准确解.

6.9

6.9 对称正定矩阵求逆及行列式的值

6.9.1 求行列式的值

在有些计算问题中，需要求出给定矩阵A的逆矩阵或行列式的值.在(6.2)中有利用Gauss消去法计算逆矩阵及行列式，矩阵A需满足顺序主子式均不为零的方法；在(6.4)中给出了矩阵求逆和行列式求值的Gauss-Jordan法，适用于A是一般非奇异矩阵。关于对称正定矩阵，求逆及行列式求值时也可应用上述两种方法，但这样做不能利用对称正定的性质，因此要有更为有利的方法.

设A是n阶对称正定矩阵．用平方根法将A分解为（见算法6.5.19）

6.9.1　$A=LL^T$

其中$L=(l_{ij})_n$为下三角矩阵．于是A的行列式之值为

6.9.2 $\det A = \det L \cdot \det L^T = \left(\prod_{i=1}^{n} l_{ii}\right)^2$

以一维数组$a[1:(n+1)n/2]$的形式，存放A的下三角部分的元素

a_{11}, a_{21}, a_{22}, a_{31}, a_{32}, a_{33}, …, a_{n1}, a_{n2}, …, a_{nn}

于是，对于$i=1, 2, \cdots, n$, 有

6.9.3 $a_{ij} = a\left[\frac{i(i-1)}{2} + j\right] \qquad (j=1,2,\cdots,i)$

6.9.4 **算法** 求对称正定矩阵A的行列式之值.结果存放在单元det中。开始det中存放1。

对于$i=1, 2, \cdots, n$

1° 计算

$p := i(i-1)/2$

2° 对于$j=1, 2, \cdots, i$

$r := j(j-1)/2$

$x := a[p+j] - \sum_{k=1}^{j-1} a[p+k]a[r+k]$

若$j<i$, 则

$a[p+j] := l_{ij} = x \times a[r+j]$

否则（即$j=i$, 这时$x=l_{ii}^2$）

$a[p+i] := 1/\sqrt{x}$（若$x \leqslant 0$则应停止计算）, $\det := x \times \det$.

6.9.5 **例** 仿照算法（6.9.4）的步骤，计算如下行列式

$$D = \begin{vmatrix} 3 & 2 & 1 \\ 2 & 2 & 1 \\ 1 & 1 & 1 \end{vmatrix}$$

解 存放D的元素的数组$a[1:6]$为

3, 2, 2, 1, 1, 1

第一步$(i=1)$: $p=0$, $j=1$, $r=0$

$x=a[1]=3$，$a[1]:=1/\sqrt{x}=1/\sqrt{3}$，$\det:=x=3$

第二步($i=2$)：$p=1$。

对于$j=1$　有$r=0$

$x=a[2]=2$，$a[2]:=x\times a[1]=2/\sqrt{3}$

对于$j=2$　有$r=1$

$x=a[3]-a[2]a[2]=2/3$，$a[3]:=1/\sqrt{x}=\sqrt{6}/2$

$\det:=x\times\det=2$

第三步($i=3$)：$p=3$。

对于$j=1$　有$r=0$

$x=a[4]=1$，$a[4]:=x\times a[1]=1/\sqrt{3}$

对于$j=2$　有$r=1$

$x=a[5]-a[4]a[2]=1/3$，$a[5]:=x\times a[3]=\sqrt{6}/6$

对于$j=3$　有$r=3$

$x=a[6]-a[4]a[4]-a[5]a[5]=1/2$

$a[6]:=1/\sqrt{x}=\sqrt{2}$，$\det:=x\times\det=1$

最后，在数组$a[1:6]$的相应位置上存放着

$1/\sqrt{3}$，$2/\sqrt{3}$，$\sqrt{6}/2$，$1/\sqrt{3}$，$\sqrt{6}/6$，$\sqrt{2}$

它们分别是

$1/l_{11}$，l_{21}，$1/l_{22}$，l_{31}，l_{32}，$1/l_{33}$。

det存放着行列式D之值，$D=1$。

6.9.2　求逆矩阵

求对称正定矩阵$A=(a_{ij})_n$的逆矩阵，A^{-1}也必对称正定。考虑关系式

6.9.6　$y=Ax$

其中$x=(x_1, x_2, \cdots, x_n)^T$，$y=(y_1, y_2, \cdots, y_n)^T$。如果从（6.9.6）能求出逆关系式

6.9.7　$x=By$

则$B=A^{-1}$即得到A的逆矩阵。

先将（6.9.6）式具体写出

$$\begin{cases} y_1=a_{11}x_1+a_{12}x_2+\cdots+a_{1n}x_n \\ y_2=a_{21}x_1+a_{22}x_2+\cdots+a_{2n}x_n \\ \cdots \\ y_n=a_{n1}x_1+a_{n2}x_2+\cdots+a_{nn}x_n \end{cases} \tag{6.9.8}$$

由于A正定，必有$a_{11}>0$，因此从（6.9.8）中的第一个等式可以解出x_1（称为交换x_1和y_1的位置）；再将得到的关于x_1的表达式代入其它等式，把（6.9.8）变换为新的关系式。为了能简便地编制程序，可将新的关系式写成如下形式：

$$\begin{cases} y_2=a'_{11}x_2+a'_{12}x_3+\cdots+a'_{1,n-1}x_n+a'_{1n}y_1 \\ y_3=a'_{21}x_2+a'_{22}x_3+\cdots+a'_{2,n-1}x_n+a'_{2n}y_1 \\ \cdots \\ y_n=a'_{n-1,1}x_2+a'_{n-1,2}x_3+\cdots+a'_{n-1,n-1}x_n+a'_{n-1,n}y_1 \\ \hline x_1=a'_{n1}x_2+a'_{n2}x_3+\cdots+a'_{n,n-1}x_n+a'_{nn}y_1 \end{cases} \tag{6.9.9}$$

容易得到新系数的计算公式为

$$\begin{cases} a'_{nn}=1/a_{11} \\ a'_{n,j-1}=-a_{1j}/a_{11} \quad (j=2,3,\cdots,n) \\ a'_{i-1,n}=a_{i1}/a_{11} \quad (i=2,3,\cdots,n) \\ a'_{i-1,j-1}=a_{ij}-a_{i1}a_{1j}/a_{11} \quad (i,j=2,3,\cdots,n) \end{cases} \tag{6.9.10}$$

注意到（6.9.9）中的x_2与y_2处于（6.9.8）中的x_1与y_1的位置，用同样的方法交换x_2与y_2的位置，又可得出新的关系式，并且只要按同样形式来写新的关系式，系数的计算公式便会保持（6.9.10）的形式。如此继续，直至交换x_n与y_n的位置，最后可求得关系式（6.9.7），实现求$A^{-1}=B$的目的。

上述过程完成前k步后，导出的关系式形如

6.9.11
$$\begin{pmatrix} y_{k+1} \\ \vdots \\ y_n \\ \hline x_1 \\ \vdots \\ x_k \end{pmatrix} = \left(\begin{array}{c|c} A_{k1} & A_{k2} \\ \hline A_{k3} & A_{k4} \end{array}\right) \begin{pmatrix} x_{k+1} \\ \vdots \\ x_n \\ \hline y_1 \\ \vdots \\ y_k \end{pmatrix}$$

其中系数矩阵分成四块，A_{k1}是$n-k$阶方阵，A_{k4}是k阶方阵，A_{k2}和A_{k3}分别是$(n-k)\times k$和$k\times(n-k)$矩阵。

可以证明：

1° $A_{k1}(1\leqslant k\leqslant n-1)$和$A_{k4}(1\leqslant k\leqslant n)$均为对称正定。因此每一步只需计算下三角（包括主对角线）元素，而且每一步计算总是可行的。

2° 对任何k有

$A_{k3}=-A_{k2}^T$

由此及A_{k1}的对称性，在计算第$k+1$步时可利用关系式：

6.9.12
$$a_{1j}=\begin{cases} a_{j1}, & j\leqslant n-k \\ -a_{j1}, & j>n-k \end{cases}$$

这里a_{1j}与a_{j1}是（6.9.11）式中系数矩阵第1行与第1列的元素。

对称正定矩阵A求逆有两个常用的算法，一个存储A的全部元素，另一个只存储A的下三角部分（包括对角线）元素。

6.9.13 **算法** 数组$a[1:n, 1:n]$，开始存放A的元素，最终存放A^{-1}的元素，下三角部分（包括对角线）是正确的。数组$h[1:n]$存放中间结果。

对于$k=1, 2, \cdots, n$：

1° 取$a[1, 1]$

$p:=a[1, 1]$

若$p\leqslant 0$则A非正定，计算停止。

2° 对于$i=2, 3, \cdots, n$

$q:=a[i, 1]$

$$h[i]:=\begin{cases} -q/p, & i\leqslant n-k+1 \\ q/p, & i>n-k+1 \end{cases}$$

$a[i-1, j-1]:=a[i, j]+q\times h[j] \qquad (j=2, \cdots, i)$

3° 计算

$a[n, n]:=1/p$

4° 对于$j=2, 3, \cdots, n$

$a[n, j-1]:=h[j]$

6.9.14 算法 数组$a[1:n(n+1)/2]$存放A的下三角部分元素

$a_{11}, a_{21}, a_{22}, a_{31}, a_{32}, a_{33}, \cdots, a_{n1}, a_{n2}, \cdots, a_{nn}$

最终存放A^{-1}的相应元素．这时a_{ij}在数组a中的位置由关系式（6.9.5）确定，另用数组$h[1:n]$存放中间结果．

对于 $k=1, 2, \cdots, n$:

1° 取$a[1]$

$p:=a[1]$

若$p\leqslant 0$则A非正定，计算停止．

2° 对于$i=2, 3, \cdots, n$

$m:=i(i-1)/2, \quad q:=a[m+1]$

$$h[i]:=\begin{cases} -q/p, & i\leqslant n-k+1 \\ q/p, & i>n-k+1 \end{cases}$$

$a[m+j-i]:=a[m+j]+q\times h[j] \quad (j=2, \cdots, i)$

3° 计算（注意：这时$m=n(n-1)/2$）

$a[m+n]:=1/p$

4° 对于$j=2, 3, \cdots, n$

$a[m+j-1]:=h[j]$

6.9.15 例 求下述矩阵的逆矩阵：

$$
A=\begin{bmatrix} 5 & 7 & 6 & 5 \\ 7 & 10 & 8 & 7 \\ 6 & 8 & 10 & 9 \\ 5 & 7 & 9 & 10 \end{bmatrix}
$$

解 应用算法(6.9.13)和(6.9.14)，两者计算结果一致。应用算法(6.9.14)的计算过程如下：

矩阵A的阶$n=4$。存放A的数组$a[1:10]$为

5, 7, 10, 6, 8, 10, 5, 7, 9, 10

第一步（$k=1$）：　$p=a[1]=5$。

对于$i=2$　有$m=1$

$q=a[2]=7$，$h[2]=-q/p=-7/5$

$a[1]:=a[3]+q\times h[2]=1/5$

对于$i=3$　有$m=3$

$q=a[4]=6$，$h[3]=-q/p=-6/5$

$a[2]:=a[5]+q\times h[2]=-2/5$

$a[3]:=a[6]+q\times h[3]=14/5$

对于$i=4$　有$m=6$

$q=a[7]=5$，$h[4]=-q/p=-1$

$a[4]:=a[8]+q\times h[2]=0$

$a[5]:=a[9]+q\times h[3]=3$

$a[6]:=a[10]+q\times h[4]=5$

然后

$a[10]:=1/p=1/5$，$a[7]:=h[2]=-7/5$

$a[8]:=h[3]=-6/5$，$a[9]:=h[4]=-1$

于是，第一步完成后，数组$a[1:10]$相应位置上出现的元素为：

1/5, −2/5, 14/5, 0, 3, 5, −7/5, −6/5, −1, 1/5

类似地可算出其余各步数组$a[1:10]$相应位置上出现的

元素，下面将它们列出。

第二步($k=2$)：

2，3，5，－4，－1，10，2，0，－7，5

第三步($k=3$)：

1/2，5，18，－3，－11，7，－3/2，－2，1，1/2

最后，第四步($k=4$)：

68，－41，25，－17，10，5，10，－6，－3，2

它们是A^{-1}的下三角部分的元素，因此A的逆矩阵

$$A^{-1}=\begin{pmatrix} 68 & -41 & -17 & 10 \\ -41 & 25 & 10 & -6 \\ -17 & 10 & 5 & -3 \\ 10 & -6 & -3 & 2 \end{pmatrix}$$

6.10

6.10 误差分析

6.10.1 解的误差估计

在求解线性代数方程组时，由于实际提供的或经过计算而得到的方程组其系数矩阵和常数项的元素总有一定误差，在将各种原始数据输入计算机并进行进制转换时会带来误差，在计算机内作每次运算还会产生舍入误差。因此，利用各种直接方法都只能求得方程组的近似解，这就需要去估计近似解的误差，或者说去估计近似解的精确度。许多例子说明，这个问题是不容忽视的。

6.10.1 例 对于方程组

$$\begin{pmatrix} 2 & 6 \\ 2 & 6.00001 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} 8 \\ 8.00001 \end{pmatrix}$$

和

$$\begin{pmatrix} 2 & 6 \\ 2 & 5.99999 \end{pmatrix}\begin{pmatrix} x_1 \\ x_2 \end{pmatrix}=\begin{pmatrix} 8 \\ 8.00002 \end{pmatrix}$$

前一方程组的解为$\boldsymbol{x}=(1,\ 1)^T$，而后一方程组的解为$\boldsymbol{x}=(10,\ -2)^T$。

这个简单的例子表明，即使两个线性代数方程组的系数及常数项极其靠近，解的差别却可能很大。也可以解释为当方程组的系数或常数项产生微小扰动（误差）时，可能使解产生很大误差，甚至失去近似的意义。这样，方程组原始数据扰动对解的影响，是要解决的第一个问题。基本线索是通过估计扰动方程组解的误差界，得出原始数据扰动对解的影响程度的一种度量——矩阵的条件数。

要解决的第二个问题是计算过程舍入误差对方程组解的影响。主要是给出关于Gauss 消去法、主元素法等舍入误差分析的结论。

对于计算机求得的方程组的近似解，原始数据扰动及舍入误差的影响不是孤立的，要完全搞清近似解的误差问题不是一件容易的事情。从理论上作出的误差界估计，又常常比实际误差大得多。因此有时还通过一些间接的简便方法来估计近似解的精确度，尽管这种估计不是对所有情形都可靠。

为了估计方程组近似解的误差，要用向量的范数和矩阵的范数。设$\boldsymbol{x}=(x_1,\ x_2,\ \cdots,\ x_n)^T$是任意的$n$维向量，$A=(a_{ij})_n$是任意的$n$阶矩阵。用记号$\|\boldsymbol{x}\|$和$\|A\|$表示向量$\boldsymbol{x}$的任何一种范数及与其相容的矩阵$A$的范数，相容是指满足条件

$$\|A\boldsymbol{x}\| \leqslant \|A\|\|\boldsymbol{x}\|$$

常用的向量范数及相容的矩阵范数如下：

1° 向量的∞-范数与矩阵的行范数为

$$\|x\|_\infty = \max_{1\leqslant i\leqslant n} |x_i|, \quad \|A\|_\infty = \max_{1\leqslant i\leqslant n} \sum_{j=1}^{n} |a_{ij}|$$

2° 向量的1－范数和矩阵的列范数为

$$\|x\|_1 = \sum_{i=1}^{n} |x_i|, \quad \|A\|_1 = \max_{1\leqslant j\leqslant n} \sum_{i=1}^{n} |a_{ij}|$$

3° 向量的2－范数和矩阵的2－范数为

$$\|x\|_2 = \left(\sum_{i=1}^{n} |x_i|^2\right)^{1/2}, \quad \|A\|_2 = [\lambda_{max}(A^TA)]^{1/2}$$

其中$\lambda_{max}(A^TA)$表示A^TA的最大特征值。

6.10.2 扰动方程组解的误差界

设有方程组

6.10.2 $Ax = b$

如果系数矩阵A和常数项b有一个扰动（微小变化或说误差），它们也可用矩阵和向量表示，记作δA和δb，，那末必然引起解x的一个扰动，记作δx。也就是说，这时实际得到的解是$x+\delta x$，它满足

6.10.3 $(A+\delta A)(x+\delta x) = b+\delta b$

分析方程组原始数据变化对解的影响（这个问题也叫做"摄动分析"），是通过δA和δb作出的对δx的某种估计。其变化大小与影响的程度可利用向量范数和矩阵范数来刻划。例如$\|\delta A\|$很小蕴涵δA每个元素的模很小，即矩阵A变化（A与$A+\delta A$的差别）微小，又如$\|\delta x\|$很大意味着δx有的元素的模很大，即对解x的影响（x与$x+\delta x$的差别）大。

6.10.4 **定义** 如果$\|\delta A\|$和$\|\delta b\|$微小，而$\|\delta x\|$却很大，则方程组（6.10.2）称为**病态方程组**（Ill-conditioned System of Equations），A称为关于解方程组或求逆的**病态矩阵**（Ill-conditioned Matrix），否则，称方程组（6.10.2）为

良态方程组，称A为良态矩阵。

例（6.10.1）中的方程组和矩阵就是病态的。但是不能笼统地说某个矩阵是“病态”的，因为一个矩阵可能对解方程组而言是病态的，但对求特征值来说却是良态的。另外，关于解方程组的病态矩阵，其病态性质是矩阵本身的属性，定义（6.10.4）中的“微小”和“很大”均系相对而言，并无数量上的严格界限。事实上，如果计算机字长加长，“病态”现象在程度上就会相对地减轻。下面说矩阵“病态”和“良态”都是对解方程组而言的。

6.10.5　定理　设A是非奇异矩阵，向量$b\neq 0$，x是方程组(6.10.2)的解，$x+\delta x$是方程组（6.10.3）的解，且矩阵A的扰动δA满足条件

$\|A^{-1}\|\|\delta A\|<1$

则有估计式

6.10.6　$$\frac{\|\delta x\|}{\|x\|}\leqslant\frac{\|A^{-1}\|\|A\|}{1-\|A^{-1}\|\|\delta A\|}\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)$$

不等式（6.10.6）反映了方程组解的相对误差与系数矩阵和常数项的相对误差的关系。在这个不等式的右端，数$\|A^{-1}\|\|A\|$起着重要的作用。

6.10.7　**定义**　设A为非奇异矩阵，则称数

$\mathrm{Cond}(A)=\|A^{-1}\|\|A\|$

为矩阵A的条件数（Condition Number）。

条件数依赖范数的选取，常用的有

$\mathrm{Cond}(A)_\infty=\|A^{-1}\|_\infty\|A\|_\infty$

$\mathrm{Cond}(A)_2=\|A^{-1}\|_2\|A\|_2$

当A是实对称矩阵时，有

$\mathrm{Cond}(A)_2=|\lambda_1|/|\lambda_2|$

其中λ_1与λ_2分别为矩阵A的绝对值最大和最小的特征值。通常，$\mathrm{Cond}(A)\geqslant 1$。

根据定义（6.10.7），不等式（6.10.6）可以改写成

6.10.8 $$\frac{\|\delta x\|}{\|x\|} \leqslant \frac{\mathrm{Cond}(A)}{1-\mathrm{Cond}(A)\frac{\|\delta A\|}{\|A\|}}\left(\frac{\|\delta A\|}{\|A\|}+\frac{\|\delta b\|}{\|b\|}\right)$$

由此可见，在条件

$$\mathrm{Cond}(A)\frac{\|\delta A\|}{\|A\|}=\|A^{-1}\|\|\delta A\|<1$$

下，对于确定的相对误差 $\|\delta A\|/\|A\|$ 与 $\|\delta b\|/\|b\|$ 来说，$\mathrm{Cond}(A)$ 越大，解的相对误差 $\|\delta x\|/\|x\|$ 可能越大；反之，$\mathrm{Cond}(A)$越小，则 $\|\delta x\|/\|x\|$ 越小。因此，$\mathrm{Cond}(A)$ 能够刻划方程组解对原始数据变化的敏感程度。$\mathrm{Cond}(A)$ 越大，矩阵A对解方程组来说就越呈病态，但条件数多大才算病态矩阵，通常并无具体标准，只是相对而言。

利用条件数可得出另一个重要结论。设$\tilde{x}$是方程组（6.10.2）的近似解，精确解x未知，欲估计$\tilde{x}$的相对误差界。令

6.10.9 $$r=b-A\tilde{x}$$

称r为剩余向量。已知$\tilde{x}$便可算出r。

6.10.10 定理 设A为非奇异矩阵，$b\neq 0$，则成立不等式

$$\frac{1}{\mathrm{Cond}(A)}\frac{\|r\|}{\|b\|}\leqslant\frac{\|\tilde{x}-x\|}{\|x\|}\leqslant\mathrm{Cond}(A)\frac{\|r\|}{\|b\|}$$

这个定理给出了方程组近似解的相对误差界。

6.10.3 病态方程组的解法

由于矩阵的条件数事先难以知道，所以实际计算需通过一些现象来判断矩阵是否病态。凡出现下列现象之一的，矩阵相对解方程组而言可能是病态的：

1° 矩阵元素间的数量级差别很大，并且无一定规律。

2° 矩阵的行列式值相对来说很小，或某些行（或列）近似地线性相关。

3° 消去过程中出现严重的有效数字丢失或模很小的主元素.

4° 剩余向量的范数已很小，但解仍不符合实际.

5° 将矩阵元素或常数项作一个小的改动，再求解一次，而解的变化很大.

对于病态方程组的求解，通常的处理办法有：适当地引入比例因子，使系数矩阵各行（或列）的元素均在±1之间，且各行（列）之范数大致相等；采用双倍（或多倍）字长进行运算；运用迭代改进解的精确度；从问题的物理背景分析产生病态的原因，纠正因问题的提法或处理不当而造成方程组病态的出现.

迭代改进的办法，是处理病态方程组求解的有效的方法之一. 只要矩阵不是非常病态，利用这个方法总可以得到相应方程组的较好的近似解. 其计算步骤如下：

1° 采取主元素法用单字长运算，将矩阵A分解为下三角矩阵L与上三角矩阵U，并将它们存放起来.

2° 用单字长运算解方程组

$L\boldsymbol{y}_0=\boldsymbol{b}$, $U\boldsymbol{x}_0=\boldsymbol{y}_0$

得出方程组$A\boldsymbol{x}=\boldsymbol{b}$的零次近似解$\boldsymbol{x}_0$.

3° 用双倍字长运算计算剩余向量

$\boldsymbol{r}_0=\boldsymbol{b}-A\boldsymbol{x}_0$

然后舍入成单字长.

4° 用单字长运算解方程组

$L\boldsymbol{y}_1=\boldsymbol{r}_0$, $U\widetilde{\boldsymbol{x}_1}=\boldsymbol{y}_1$

求出解的修正量$\widetilde{\boldsymbol{x}_1}$.

5° 计算修正后的解

$\boldsymbol{x}_1=\boldsymbol{x}_0+\widetilde{\boldsymbol{x}}$

6° 将$\boldsymbol{x}_1$代替$\boldsymbol{x}_0$，重复3°—5°.

如此反复，可得出近似解的序列

$x_0,\ x_1,\ x_2,\ \cdots,\ x_k,\ \cdots$

如果k大于某个k_0后，所有x_k都近似地等于某个单字长向量x^*，则除个别情况外，x^*即为真解$A^{-1}b$的正确舍入解.

以上迭代改进过程只将矩阵A分解一次，计算工作量约为$n^3/3$，其后每次迭代主要是解两个三角形方程组，计算工作量为n^2. 对于不十分病态的矩阵，一般只需迭代4～5次即可. 所以，因迭代而增加的计算工作量并不多.

当矩阵A十分病态时，上述迭代过程产生的序列$\{x_k\}_{k=1}^{\infty}$不收敛，必须采取用多字长运算等其它措施.

6.10.4 舍入误差

设x和$\widetilde{x}$分别是方程组（6.10.2）的精确解和计算解. 跟踪计算过程逐步分析舍入误差来获得误差$\|x-\widetilde{x}\|$的界，称为**向前误差分析法**（Forward Error Analysis）. 显然，使用这种方法会遇到很大的困难. 目前经常采用的分析舍入误差的一种方法是所谓**向后误差分析法**（Backward Error Analysis）. 其基本思想是把计算过程舍入误差对解的影响归结为原始数据扰动对解的影响. 具体地说，要寻找原始数据的某种扰动δA和δb，使得计算解$\widetilde{x}$严格地满足

$$(A+\delta A)\widetilde{x}=b+\delta b$$

找到这样的δA和δb后，便可利用定理（6.10.5）的结论估计误差$\|x-\widetilde{x}\|$的界.

对于不同的算法，δA、δb将会不同. 而且从（6.10.6）可知，为了估计$\widetilde{x}$的相对误差，只需知道$\|\delta A\|$和$\|\delta b\|$. 各种直接法已进行过详细的误差分析，得出了相应的$\|\delta A\|$和$\|\delta b\|$的上界. 下面列出与几个主要算法有关的结果，其中n是矩阵A的阶数，t是计算机的字长，a是A的按模最大的

元素，即

$$a=\max_{1\leqslant i,\,j\leqslant n}|a_{ij}|$$

1° Gauss消去法

6.10.11 $\|\delta A\|_\infty\leqslant c_1Ga(2n^2+n^3)2^{-t},\quad \|\delta b\|_\infty=0$

其中c_1为接近于1的某个常数，G为消去过程中矩阵元素的最大增长因子，即

$$G=\max_{\substack{1\leqslant i,\,j\leqslant n\\ 0\leqslant k\leqslant n-1}}|a_{ij}^{(k)}|/a$$

对于列主元素消去法有

$$G\leqslant 2^{n-1}$$

增长因子增长很快，达到界限2^{n-1}的矩阵确实是存在的，但这种情形极少见。

对于全主元消去法有

$$G\leqslant(n\cdot 2^1\cdot 3^{1/2}\cdot 4^{1/3}\cdot\cdots\cdot n^{1/(n-1)})^{1/2}$$

这时$G\leqslant 2n^{(\log n)/4+1/2}$，因此增长因子增长缓慢。

将（6.10.11）代入（6.10.8），并利用条件

6.10.12 $\alpha=\|A^{-1}\|\,\|\delta A\|<1$

便得计算解$\widetilde{x}$的相对误差界

6.10.13 $$\frac{\|x-\widetilde{x}\|_\infty}{\|x\|_\infty}\leqslant \mathrm{Cond}(A)\left(\frac{c_1}{1-\alpha}\cdot G\right)(2n^2+n^3)2^{-t}$$

2° 直接三角分解法。结果完全与消去法类似。

3° 平方根法

6.10.14 $\|\delta A\|_\infty\leqslant c_2a(2n^2+n^3)2^{-t},\quad \|\delta b\|_\infty=0$

计算解x的相对误差界为

$$\frac{\|x-\widetilde{x}\|_\infty}{\|x\|_\infty}\leqslant \mathrm{Cond}(A)\left(\frac{c_2}{1-\alpha}\right)(2n^2+n^3)2^{-t}$$

以上c_2是接近于1的常数。

从这些估计式可以看出，矩阵A的条件数越大，阶数越

高，计算机字长越短，则舍入误差时解的影响越严重。当然，这些估计式都是严格上界，往往是保守的。例如，经验证明对于消去法将有如下结果：

$$\frac{\| x-\tilde{x} \|_\infty}{\| x \|_\infty} \leqslant cGn2^{-t}\mathrm{Cond}(A) \tag{6.10.15}$$

这显然比（6.10.13）中的上界小得多。

6.10.5 近似解精确度的检验

实际使用近似解的误差估计式有一定困难，因此，在实际计算中，当得出近似解$\tilde{x}$后，有两种经常使用的估计精确度的办法。

一种是将近似解$\tilde{x}$代入原方程组（6.10.2），算出剩余向量$r=b-A\tilde{x}$。如果r的每个分量r_i的模与常数项b的相应分量b_i的模相比较都是小量（当$b_i=0$时r_i应接近于零），则一般认为$\tilde{x}$是相当准确的，否则认为$\tilde{x}$是不准确的。这种方法简单、运算量少，对大多数实际问题是很可靠的。缺点是从剩余向量只能得出近似解准确与否的粗略概念，而无法断定近似解究竟有几位是准确的。此外，这种方法对病态方程组是不可靠的，因为这时可能出现剩余向量已接近零向量，但近似解误差很大，甚至完全不准确。例如，在例（6.10.1）中，将$x=(10,\ -2)^T$代入第一个方程组，将$x=(1,\ 1)^T$代入第二个方程组，得到的剩余向量都是

$$r=\begin{pmatrix} 0 \\ 0.00003 \end{pmatrix}$$

非常接近零向量。而实际上$x=(1,\ 1)^T$是第一个方程组的准确解，$x=(10,-2)^T$是第二个方程组的准确解。

另一种办法是任取一个已知向量z，用较多的位数计算

出向量Az，并以其为常数项，按求出方程组（6.10.2）近似解$\tilde{x}$的同样方法求解方程组

$$Ax=Az$$

所得的解记作x'。如计算过程无舍入误差，x'应等于z。实际上则因求解过程中舍入误差的积累，x'与z之间将有差异。由于x'与$\tilde{x}$的求解过程完全相同，系数矩阵也一样，一般认为两者的精确度是相同的。可把x'与z各分量之间相符合的最少位数作为近似解$\tilde{x}$的有效位数。这种方法比较可靠，能得到近似解有几位准确的一个数量概念，缺点是计算量较大。若是先算出Az，然后与b并列求解，则可节省运算量。

7

第7章 解线性方程组的迭代法

7.1

7.1 引 言

对于具有极少量非零元素的大型稀疏矩阵方程组，采用迭代方法比较适宜。迭代方法的程序设计简单，由于只需要存储非零元素或者说可以采用压缩存储技巧，所以节省存储单元。对于许多问题，例如二阶椭圆型边值问题的差分方程组的求解问题，迭代法收敛较快，但是，对于某些问题，迭代法可能发散或收敛很慢，因此，迭代法存在着收敛和收敛速度问题。

迭代法的使用与问题的特点有着密切的关系，使用时应根据实际问题选用合适的迭代方法，既要有一定的经验还应有理论的分析和指导。

7.2

7.2 范数、序列极限、条件数

7.2.1 向量范数

度量n维列向量空间R^n中向量$x=(x_1, x_2, \cdots x_n)^T$的"大小"以及度量$R^n$中两个向量$x$和$y$之间的"距离"，可以使用向量范数（Vector Norm）。

7.2.1 **定义** 设 $\|\cdot\|$ 是R^n上的一个实值函数，满足下述三个性质：

1° $\|x\|\geqslant 0$，当且仅当$x=0$时 $\|x\|=0$(正定条件)；

2° 对任意一个实数，$\|cx\|=|c|\,\|x\|$（齐次条件）；

3° $\|x+y\|\leqslant\|x\|+\|y\|$（三角不等式）

称 $\|x\|$ 为R^n上的一个**向量范数**。

由3°可推出不等式

$$\left|\|x\| - \|y\|\right| \leqslant \|x-y\|$$

7.2.2 定义 设$x=(x_1, x_2, \cdots x_n)^T\in R^n$，则向量$x$的三种常用范数定义为：

1° 向量的"1"范数

$$\|x\|_1=|x_1|+|x_2|+\cdots+|x_n|=\sum_{i=1}^{n}|x_i|$$

2° 向量的"∞"范数（或称最大范数）

$$\|x\|_\infty=\max(|x_1|, |x_2|, \cdots, |x_n|)=\max_{1\leqslant i\leqslant n}|x_i|$$

3° 向量的"2"范数（或称欧几里得范数）

$$\|x\|_2=\sqrt{|x_1|^2+|x_2|^2+\cdots+|x_n|^2}=\left(\sum_{i=1}^{n}|x_i|^2\right)^{1/2}$$

$$=(x, x)^{1/2}$$

这样定义的向量x的函数$\|\cdot\|_1$，$\|\cdot\|_\infty$和$\|\cdot\|_2$满足定义（7.2.1）的三个条件，因此它们都是R^n上的向量范数。

7.2.3 定义 设$x\in R^n$，则

$$\|x\|_p=\left(\sum_{i=1}^{n}|x_i|^p\right)^{1/p}, \quad 1\leqslant p<\infty$$

称为向量x的Hölder（**荷尔德**）**范数**或**P-范数**。

向量x的范数$\|x\|_1$，$\|x\|_2$和$\|x\|_\infty$都是p-范数的特殊情况（$\|x\|_\infty$是$\lim\limits_{p\to\infty}\|x\|_p$）。

7.2.4 例 计算向量$x=(3, 0, -4, -12)^T$的三种范数。

解 $\|x\|_1=3+4+12=19$

$\|x\|_\infty=\max(3, 4, 0, 12)=12$

$\|x\|_2=(3^2+4^2+12^2)^{1/2}=13$

7.2.5 定理/ $\|\cdot\|$ 的连续性 设非负函数$\|\cdot\|$为R^n上任一向量范数，则$\|\cdot\|$是x分量x_1，x_2，…，x_n的连续函数。

7.2.6 定理/向量范数的等价性 设 $\|x\|_s$, $\|x\|_t$ 为 R^n 上向量的任意两种范数，则存在常数$c_1>0$，$c_2>0$，使得对一切$x\in R^n$有

$$c_1\|x\|_s\leqslant\|x\|_t\leqslant c_2\|x\|_s$$

7.2.7 例 对于每一个向量$x\in R^n$有

$$\|x\|_\infty\leqslant\|x\|_2\leqslant\sqrt{n}\|x\|_\infty$$

$$\|x\|_\infty\leqslant\|x\|_1\leqslant n\|x\|_\infty$$

7.2.2 矩阵范数

可以用矩阵范数来度量两个n阶矩阵之间的"距离"以及研究矩阵序列的收敛性。

7.2.8 定义 如果矩阵$A\in R^{n\times n}$的某个非负的实值函数 $\|\cdot\|$ 满足下述条件：

1° $\|A\|\geqslant0$，当且仅当$A=0$时 $\|A\|=0$（正定条件）；

2° 对任意实数α，有

$\|\alpha A\|=|\alpha|\,\|A\|$（齐次条件）；

3° $\|A+B\|\leqslant\|A\|+\|B\|$（三角不等式）；

4° $\|AB\|\leqslant\|A\|\,\|B\|$

则称 $\|A\|$ 是 $R^{n\times n}$上的一个**矩阵范数**（Matrix Norm）。

7.2.9 定义 设$x\in R^n$，$A\in R^{n\times n}$，给出一种向量范数$\|x\|_v$（如$v=1$，2或∞），相应地定义一个矩阵的非负函数

$$\|A\|_v=\max_{x\neq0}\frac{\|Ax\|_v}{\|x\|_v}$$

可验证 $\|A\|_v$满足定义（7.2.8）的条件，所以 $\|A\|_v$是$R^{n\times n}$上矩阵的一个范数，称为A的**算子范数**（Operator Norm）或称为A的**自然范数**（Natural Norm）。

7.2.10 定理 设$x\in R^n$，$A\in R^{n\times n}$，$\|A\|$ 是矩阵A的算子范数，则

$$\|A\| = \max_{\|x\|=1} \|Ax\|$$

7.2.11 定理 设$\|x\|_\nu$是R^n上的一个向量范数，则$\|A\|_\nu$是$R^{n\times n}$上矩阵的范数，且满足相容条件

$$\|Ax\|_\nu \leqslant \|A\|_\nu \|x\|_\nu \quad (\nu=1, 2, \infty, \cdots)$$

7.2.12 定理 设$x\in R^n$，$A\in R^{n\times n}$，则

1° $\|A\|_\infty = \max\limits_{1\leqslant i\leqslant n} \sum\limits_{j=1}^{n} |a_{ij}|$（称为$A$的**行范数**）

2° $\|A\|_1 = \max\limits_{1\leqslant j\leqslant n} \sum\limits_{i=1}^{n} |a_{ij}|$（称为$A$的**列范数**）

3° $\|A\|_2 = \sqrt{\lambda_{max}(A^TA)}$（称为$A$的**2-范数，或谱范数**）

其中$\lambda_{max}(A^TA)$表示A^TA的最大特征值。

由此可知，计算一个矩阵的$\|A\|_\infty$，$\|A\|_1$还是比较容易的，而矩阵的2-范数$\|A\|_2$在计算上不方便，但它在理论上是有用的。

7.2.13 定义 设$A\in R^{n\times n}$

$$\|A\|_F = \left(\sum_{i,j=1}^{n} a_{ij}^2\right)^{1/2}$$

则称$\|A\|_F$为A的**F范数**（Frobenius范数），它是一种与向量范数不相关的矩阵范数。$\|A\|_F$显然满足正定性、齐次性及三角不等式。

7.2.14 例 计算单位矩阵I_n的四种矩阵范数（n为矩阵的阶）。

解 $\|I_n\|_1=1$，$\|I_n\|_\infty=1$，$\|I_n\|_2=1$，$\|I_n\|_F=\sqrt{n}$。从此例可以看出，从属于任一种向量范数的单位矩阵的范数皆是1，而不从属于向量范数的单位矩阵的F范数不是1。

7.2.15 例 设$A=\begin{bmatrix} 1 & -2 \\ -3 & 4 \end{bmatrix}$，试计算$A$的各种范数。

解 $\|A\|_1=6$，$\|A\|_\infty=7$，$\|A\|_F\approx 5.477$

$\| A \|_2=\sqrt{15+\sqrt{221}}\approx 5.46$

7.2.16 定理 设$A\in \boldsymbol{R}^{n\times n}$，则

1° $\| A^T \|_2=\| A \|_2$

2° $\| A^TA \|_2=\| A \|_2^2$

3° 如果U为正交矩阵，那么$\| UA \|_2=\| A \|_2$。

7.2.17 定义/谱半径 设$A\in \boldsymbol{R}^{n\times n}$的特征值为$\lambda_i(i=1, 2, \cdots, n)$，称

$$\rho(A)=\max_{1\leqslant i\leqslant n} |\lambda_i|$$

为A的**谱半径**(Spectral Radius)

7.2.18 定理 设$A\in \boldsymbol{R}^{n\times n}$，则

1° $[\rho(A^TA)]^{1/2}=\| A \|_2$

2° $\rho(A)\leqslant\| A \|$

其中，2°说明A的谱半径不超过A的任何一种范数，即$\| A \|$是A的特征值的上界。

7.2.19 定理 如果$A\in \boldsymbol{R}^{n\times n}$为对称矩阵，则$\| A \|_2=\rho(A)$。

7.2.20 定理 如果$\| B \|<1$，则$I\pm B$为非奇异矩阵，且

$$\| (I\pm B)^{-1} \|\leqslant\frac{1}{1-\| B \|},$$

其中 $\|\cdot\|$ 是指矩阵的算子范数。

7.2.21 定理/矩阵算子范数的等价性 设$\| A \|_r$，$\| A \|_s$为任意两种$\boldsymbol{R}^{n\times n}$上的矩阵算子范数，则存在常数$c_1$，$c_2>0$，使

$c_1\| A \|_r\leqslant\| A \|_s\leqslant c_2\| A \|_r$

其中$A\in \boldsymbol{R}^{n\times n}$。

7.2.3 序列极限

用向量范数和矩阵范数来描述**向量序列极限**(Vector Sequence Limit)和**矩阵序列极限**(Matrix Sequence Limit)。

7.2.22 定义 设$\{x^{(k)}\}$为R^n中一向量序列(Vector Sequence)，$x^* \in R^n$，记$x^{(k)}=(x_1^{(k)}, x_2^{(k)}, \cdots, x_n^{(k)})^T$，$x^*=(x_1^*, x_2^*, \cdots, x_n^*)^T$，如果$\lim\limits_{k\to\infty} x_i^{(k)}=x_i^*(i=1, 2, \cdots n)$，则称序列$\{x^{(k)}\}$收敛于向量$x^*$，记为

$$\lim_{k\to\infty} x^{(k)}=x^*$$

向量范数是该向量到原点之间距离的一种度量，因而两个向量之差的范数是它们之间距离的一种度量。

7.2.23 定义 设有两个向量$x=(x_1, x_2, \cdots, x_n)^T$和$y=(y_1, y_2, \cdots, y_n)^T$，它们之间的"距离"定义为$\| x-y \|$。例如

$$\| x-y\|_2=\left\{\sum_{i=1}^{n} |x_i-y_i|^2\right\}^{1/2}$$

和

$$\| x-y\|_\infty=\max_{1\leqslant i\leqslant n} |x_i-y_i|$$

7.2.24 例 线性方程组

$$\begin{cases} 3.3330x_1+15920x_2-10.333x_3=15913 \\ 2.2220x_1+16.710x_2+9.6120x_3=28.544 \\ 1.5611x_1+5.1791x_2+1.6852x_3=8.4254 \end{cases}$$

有解$x=(x_1, x_2, x_3)^T=(1.0000, 1.0000, 1.0000)^T$，若用列主元消去法求解，用5位有效数字进行运算，得到解$\widetilde{x}=(\widetilde{x}_1, \widetilde{x}_2, \widetilde{x}_3)$，$\widetilde{x}=(1.2001, 0.99991, 0.92538)^T$，试求$\widetilde{x}$与$x$之间的误差（可用这两个向量之间的距离来度量）。

解

$$\begin{aligned} \| x-\widetilde{x} \|_\infty &=\max\{ |1.0000-1.2001|, |1.0000-0.99991|, |1.0000-0.92538| \} \\ &=\max\{0.2001, 0.00009, 0.07462\} \\ &=0.2001 \end{aligned}$$

$$\| x-\widetilde{x} \|_2=(|1.0000-1.2001|^2+|1.0000-0.99991|^2+|1.0000-0.925381|^2)^{1/2}$$

$$=(0.2001^2+0.00009^2+0.07462^2)^{1/2}$$
$$=0.21356$$

7.2.25　定理　$\lim\limits_{k\to\infty} x^{(k)}=x^* \Longleftrightarrow \|x^{(k)}-x^*\|\to 0$　（当$k\to\infty$时），

其中 $\|\cdot\|$ 为向量的任一种范数。

7.2.26　定义　设$A=(a_{ij})_n$，如果

$$\lim_{k\to\infty}(A^k)_{ij}=0 \quad (i, j=1,\ 2, \cdots n)$$

则称矩阵A是收敛的（Convergent）。

7.2.27　例　设 $A=\begin{pmatrix}\frac{1}{2} & 0\\ \frac{1}{4} & \frac{1}{2}\end{pmatrix}$，计算$A$的幂。

$$A^2=\begin{pmatrix}\frac{1}{4} & 0\\ \frac{1}{4} & \frac{1}{4}\end{pmatrix},\quad A^3=\begin{pmatrix}\frac{1}{8} & 0\\ \frac{3}{16} & \frac{1}{8}\end{pmatrix},\quad A^4=\begin{pmatrix}\frac{1}{16} & 0\\ \frac{1}{8} & \frac{1}{16}\end{pmatrix},$$

$$\cdots\cdots A^k=\begin{pmatrix}\left(\frac{1}{2}\right)^k & 0\\ \frac{k}{2^{k+1}} & \left(\frac{1}{2}\right)^k\end{pmatrix},\ \cdots\cdots$$

因为$\rho(A)=\frac{1}{2}<1$，则$\lim\limits_{k\to\infty}\left(\frac{1}{2}\right)^k=0$ 和 $\lim\limits_{k\to\infty}\frac{k}{2^{k+1}}=0$，所以按定义（7.2.26），$A$是收敛的。

7.2.28　定理　$\lim\limits_{k\to\infty}(A^k)_{ij}=0 \Longleftrightarrow \rho(A)<1$

7.2.29　定理　$\lim\limits_{k\to\infty}(A^k)_{ij}=0 \Longleftrightarrow \lim\limits_{k\to\infty}\|A^k\|=0$

7.2.30　定义　设有矩阵序列$A^{(k)}=(a_{ij}^{(k)})_n$（$k=1,\ 2,\ \cdots$）及

$A=(a_{ij})_n$，如果

$$\lim_{k\to\infty} a_{ij}^{(k)}=a_{ij} \quad (i,j=1,2,\cdots n)$$

则$\lim\limits_{k\to\infty} A^{(k)}=A$，称矩阵序列$\{A^{(k)}\}$收敛于$A$。

7.2.31 定理 $\lim\limits_{k\to\infty} A^{(k)}=A \Longleftrightarrow \lim\limits_{k\to\infty} \|A^{(k)}-A\|$

$=0 \Longleftrightarrow$对任意x有 $\lim\limits_{k\to\infty} A^{(k)}x=Ax$

7.2.32 定理 设$\lim\limits_{k\to\infty} A^{(k)}=A$和 $\lim\limits_{k\to\infty} B^{(k)}=B$，于是有

1° $\lim\limits_{k\to\infty} (A^{(k)}+B^{(k)})=A+B$

2° $\lim\limits_{k\to\infty} A^{(k)}B^{(k)}=AB$

3° 如果A是非异的，则对一切充分大的k，$A^{(k)}$是非奇异的，且$\lim\limits_{k\to\infty}(A^{(k)})^{-1}=A^{-1}$

7.2.4 矩阵的条件数

7.2.33 定义/矩阵的条件数 设A为非奇异矩阵，称数

$$\mathrm{Cond}(A)=\|A^{-1}\|\,\|A\|$$

为矩阵A的条件数（Condition Number）。

通常使用的条件数有

7.2.34 $\mathrm{Cond}(A)_\infty=\|A^{-1}\|_\infty\|A\|_\infty$

7.2.35 **A的谱条件数**（Spectral Condition Number）

$$\mathrm{Cond}(A)_2=\|A^{-1}\|_2\|A\|_2=\sqrt{\frac{\lambda_{\max}(A^TA)}{\lambda_{\min}(A^TA)}}$$

当A为对称正定时，

$$\mathrm{Cond}(A)_2=\frac{\lambda_1}{\lambda_n}$$

其中λ_1，λ_n为A的最大及最小特征值。

7.2.36 矩阵A的条件数的性质：

1° 对任何非奇异矩阵A，必有$\text{Cond}(A)\geqslant 1$，即

$$\text{Cond}(A)=\|A^{-1}\|\,\|A\|\geqslant\|A^{-1}A\|=\|I_n\|\geqslant 1$$

2° 设A为非奇异矩阵，且$c\neq 0$（常数），则

$$\text{Cond}(cA)=\text{Cond}(A)$$

3° 如果A为正交矩阵，则$\text{Cond}(A)_2=1$

4° 如果A为非奇异阵，R为正交矩阵，则

$$\text{Cond}(RA)_2=\text{Cond}(AR)_2=\text{Cond}(A)_2$$

5° $\text{Cond}(A\cdot B)\leqslant\text{Cond}(A)\cdot\text{Cond}(B)$

7.2.37 **定义** 设$Ax=b$，其中A为非奇异矩阵。如果A的条件数比1大得多，即$\text{Cond}(A)\gg 1$，则称A**是坏条件的**或 **A 为病态的**（Ill-conditioned Matrix），称方程组 $Ax=b$**是坏条件的**或称**方程组是病态的**(Ill-conditioned Equations)。如果 A 的条件数相对地小，则称A**是好条件的**或**A是良态的**（Well-conditioned Matrix），称方程组$Ax=b$ **是好条件的**或称**方程组是良态的**（Well-conditioned Equations）。

7.2.38 **例** 计算矩阵$A=\begin{pmatrix}1 & 1\\ 1 & 1.0001\end{pmatrix}$的$\text{Cond}(A)_\infty$。

解 先计算A^{-1}

$$A^{-1}=\begin{pmatrix}10001 & -10000\\ -10000 & 10000\end{pmatrix}$$

于是 $\text{Cond}(A)_\infty=\|A^{-1}\|_\infty\|A\|_\infty=20001\times 2.0001=40004$

故A是病态的。

7.2.39 **例** 当方程组

$$\begin{pmatrix}1 & 1\\ 1 & 1.0001\end{pmatrix}\begin{pmatrix}x_1\\ x_2\end{pmatrix}=\begin{pmatrix}2\\ 2\end{pmatrix}$$

的右端项$b=(2,\ 2)^T$的第2个分量有1/10000的变化时，方程组的解有多大变化？

解 此方程组的右端项在变化前的解是$x=(2,\ 0)^T$，变化后成为

$$\begin{pmatrix} 1 & 1 \\ 1 & 1.0001 \end{pmatrix}\begin{pmatrix} x_1+\delta x_1 \\ x_2+\delta x_2 \end{pmatrix}=\begin{pmatrix} 2 \\ 2.0001 \end{pmatrix}$$

其解$x+\delta x=(1,\ 1)^T$。

由此可以看出，方程组右端项的微小变化引起解的很大的变化，所以用矩阵的条件数完全可以刻划方程组或矩阵的病态程度。

7.2.40 **例** 对于Hilbert（希尔伯特）矩阵

$$H_n=\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} & \cdots & \frac{1}{n} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} & \cdots & \frac{1}{n+1} \\ \cdots \\ \frac{1}{n} & \frac{1}{n+1} & \frac{1}{n+2} & \cdots & \frac{1}{2n-1} \end{pmatrix}$$

当$n=3$时，计算H_3的条件数。

解

$$H_3=\begin{pmatrix} 1 & \frac{1}{2} & \frac{1}{3} \\ \frac{1}{2} & \frac{1}{3} & \frac{1}{4} \\ \frac{1}{3} & \frac{1}{4} & \frac{1}{5} \end{pmatrix},\ H_3^{-1}=\begin{pmatrix} 9 & -36 & 30 \\ -36 & 192 & -180 \\ 30 & -180 & 180 \end{pmatrix}$$

于是

$$\|H_3\|_\infty=\frac{11}{6},\ \|H_3^{-1}\|_\infty=408,\quad \mathrm{Cond}(H_3)_\infty=748$$

故A为病态矩阵。

当$n=6$时，算得$\mathrm{Cond}(H_6)=2.9\times10^6$。对于一般的$H_n$矩阵，当$n$愈大时病态愈严重。

7.3 矩阵理论

7.3.1 不可约性和对角占优矩阵

7.3.1 **定义** 设$A\in R^{n\times n}$，如果存在排列矩阵P使得

7.3.2 $$PAP^{-1}=\begin{pmatrix}F & O\\ G & H\end{pmatrix}$$

其中F和H是方阵，O是零矩阵，则称A为**可约矩阵**（Reducible Matrix）（或**可分矩阵**），否则称A为**不可约矩阵**（Irreducible Matrix）（或**不可分矩阵**）。

解方程组

$Ax=b$

时，如果A是非奇异的，但可约，那末可将未知量的次序和方程的次序重新排列，得到矩阵的形式如（7.3.2）。这样便可单独求解矩阵F和H的方程组。

7.3.3 **定义** 设$A=(a_{ij})_n\in R^{n\times n}$，如果

7.3.4 $$|a_{i,i}|\geqslant\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|\qquad(1\leqslant i\leqslant n)$$

且至少对一个i有

7.3.5 $$|a_{i,i}|>\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{i,j}|$$

则称A是**弱对角占优的**（Weakly Diagonally Dominant）。如果严格不等式（7.3.5）对$1\leqslant i\leqslant n$都成立，则称A是**严格对角占优的**（Strictly Diagonally Dominant）。

7.3.6 **定理** 如果A是弱对角占优的不可约矩阵，则$\det A\neq 0$且$a_{ii}\neq 0(i=1,2,\cdots,n)$。

7.3.7 定理 如果A是严格对角占优矩阵，则$\det A\neq0$。若a_{ii}是正的实数，则A的特征值满足$\lambda_i>0$（$1\leqslant i\leqslant n$）。

7.3.8 例 $A=\begin{pmatrix}2&-1&0\\-1&2&-1\\0&-1&2\end{pmatrix}$

A是弱对角占优的不可约矩阵，$\det A=4\neq0$

7.3.9 例 $A=\begin{pmatrix}4&-1&0\\-1&4&-1\\0&-1&4\end{pmatrix}$

A是严格对角占优矩阵，$\det A=56\neq0$。因为$a_{ii}=4>0(i=1,2,3)$，故可求得$\lambda_1=4+\sqrt{2}$，$\lambda_2=4$，$\lambda_3=4-\sqrt{2}$皆大于零。

7.3.2 对称正定矩阵

7.3.10 定义 设$A\in R^{n\times n}$，$(\cdot,\ \cdot)$是向量内积，如果A对称且对任何非零向量v有

$(Av,\ v)>0$

则称A是**对称正定的**(Symmetric and Positive Definite)。

7.3.11 定理 如果A是$n\times n$的正定矩阵，则$\det A\neq0$。

7.3.12 定理 正定矩阵是特征值都大于零的实对称矩阵。

7.3.13 定理 A是对称正定矩阵的充分必要条件为：A的主子式$A_i>0$ （$i=1,\ 2,\ \cdots,\ n$）。

7.3.14 定理 如果A是$n\times n$的严格对角占优的对称矩阵，或者是对角线元素均为正实数的不可约弱对角占优矩阵，则A是正定的。

7.3.3 性质'A'和相容次序

7.3.15 定义 如果矩阵A具有形式

7.3.16
$$A=\begin{pmatrix} D_1 & U_1 & & & \\ L_2 & D_2 & U_2 & & \\ & L_3 & D_3 & U_3 & \\ & & \ddots & \ddots & \ddots \\ & & & \ddots & \ddots & U_{m-1} \\ & & & & L_m & D_m \end{pmatrix}$$

其中D_i为对角线型方阵，L_i，U_i为相应阶数的长方阵，则称A为**D-型分块三对角矩阵** (Diagonal Form of Partitioned Tridiagonal Matrix)。

7.3.17 例

$$A_1=\left(\begin{array}{ccc|ccc|ccc} a_{11} & a_{12} & 0 & a_{14} & & & & & \\ a_{21} & a_{22} & a_{23} & 0 & a_{25} & & & & \\ & a_{32} & a_{33} & 0 & 0 & a_{36} & & & \\ \hline a_{41} & 0 & 0 & a_{44} & a_{45} & 0 & a_{47} & & \\ & a_{52} & 0 & a_{54} & a_{55} & a_{56} & 0 & a_{58} & \\ & & a_{63} & 0 & a_{65} & a_{66} & 0 & 0 & a_{69} \\ \hline & & & a_{74} & 0 & 0 & a_{77} & a_{78} & 0 \\ & & & & a_{85} & 0 & a_{87} & a_{88} & a_{89} \\ & & & & & a_{96} & 0 & a_{98} & a_{99} \end{array}\right)$$

$$=\begin{pmatrix} D_1 & U_1 & \\ L_2 & D_2 & U_2 \\ & L_3 & D_3 \end{pmatrix}$$

A_1是D-型分块三对角阵。

7.3.18 **定义/性质'A'** 设$A\in R^{n\times n}$，如果存在一个排列矩阵P使得

7.3.19 $$PAP^{-1}=\begin{pmatrix} D_1 & U_1 \\ L_2 & D_2 \end{pmatrix}$$

其中D_1，D_2为两个对角线阵，则称矩阵A具有**性质'A'**(Property'A')。

性质'A'的更确切的定义如下：

7.3.20 **定义/性质'A'** 设$A\in R^{n\times n}$，若能将前n个正整数所构成的集合$W=\{1,2,\cdots n\}$分成两个不相交的子集合S_1，S_2（即$S_1+S_2=W$，S_1与S_2无公共元素），使得矩阵A对角线以外的每一个非零元素$a_{ij}(i\neq j)$的足标对(i,j)均满足$i\in S_1$，$j\in S_2$或$i\in S_2$，$j\in S_1$，则称此矩阵具有**性质'A'**。

7.3.21 **例** 用定义(7.3.20)验证例(7.3.17)中的D-型三对角矩阵A_1具有**性质'A'**。

解 因为A_1是九阶矩阵，前九个正整数构成集合$W=\{1,2,3,\cdots,9\}$，把它分成二个不相交的子集$S_1=\{1,3,5,7,9\}$，$S_2=\{2,4,6,8\}$，$S_1+S_2=W$，S_1和S_2无公共元素。对于任一个a_{ij} $(i\neq j)$的足标对(i,j)均满足$i\in S_1$，$j\in S_2$或$i\in S_2$，$j\in S_1$，故例(7.3.17)中的矩阵A_1具有**性质'A'**。

7.3.22 **例** 用定义(7.3.18)验证(7.3.17)中的D-型三对角矩阵A_1具有性质'A'。

解 将A_1中编号属于集合S_1的各行（以及相应的各列）均排至前面，而将属于集合S_2的均排在后面，可把矩阵A_1变成(7.3.19)的矩阵形式。

将A_1按例(7.3.21)中的次序S_1，S_2重新排列后得

7.3.23
$$PA_1P^{-1}=\left(\begin{array}{ccccc|cccc}
a_{11} & & & & & a_{12} & a_{14} & & \\
 & a_{33} & & & & a_{32} & 0 & a_{36} & \\
 & & a_{55} & & & a_{52} & a_{54} & a_{56} & a_{58} \\
 & & & a_{77} & & & a_{74} & 0 & a_{78} \\
 & & & & a_{99} & & & a_{96} & a_{98} \\
\hline
a_{21} & a_{23} & a_{25} & & & a_{22} & & & \\
a_{41} & 0 & a_{45} & a_{47} & & & a_{44} & & \\
 & a_{63} & a_{65} & 0 & a_{69} & & & a_{66} & \\
 & & a_{85} & a_{87} & a_{89} & & & & a_{88}
\end{array}\right)$$

$$=\begin{pmatrix} D_1 & U_1 \\ L_2 & D_2 \end{pmatrix}$$

其中P是排列阵，D_1和D_2是对角线阵，故由定义（7.3.18），矩阵A_1具有**性质'A'**。

很容易验证形如（7.3.16）的D–型三对角矩阵具有**性质'A'**。在矩形网格上用五点差分格式逼近二阶椭圆型微分方程所得到的系数矩阵是D–型块三对角阵，它具有**性质'A'**。

7.3.24 定义/相容次序 若能将前n个正整数所构成的集合 $\boldsymbol{W}$ 分成t个不相交的子集$S_1, S_2, \cdots, S_t$（即$\sum\limits_{k=1}^{t} S_k=\boldsymbol{W}$，$S_i$与$S_j$（$i\neq j$）无公共元素），使得矩阵$A$的任意非对角非零元素$a_{ij}\neq 0$（$i\neq j$）的足标对$(i, j)$满足如下条件，若$i\in S_k$时$j\in S_{k-1}$（当$j<i$）或$j\in S_{k+1}$（当$j>i$），则称此矩阵具有**相容次序**（Consistently Ordered）。

将（7.3.16）中属于对角线子块D_k之行编号记为集合

S_k，显然，$\sum_{k=1}^{m} S_k = W$且S_i与$S_j(i \neq j)$无公共元素.(7.3.17)中矩阵A_1内的任意非对角线非零元素$a_{ij} \neq 0(i \neq j)$，其足标对(i,j)均满足定义(7.3.24)中的条件，所以形如(7.3.16)的矩阵具有相容次序。

7.3.4 M-矩阵与正则分解

7.3.25 定义 设$A \in R^{n \times n}$，如果A非奇异，$A^{-1} \geq 0$且

$a_{ij} \leq 0 \quad (i \neq j, \quad i,j=1,2,\cdots N)$

则称A为M-矩阵。

7.3.26 定理 设$A \in R^{n \times n}$，满足

$a_{ii} > 0 \quad (i=1,2,\cdots,N)$

$a_{ij} \leq 0 \quad (i \neq j, \quad i,j=1,2,\cdots N)$

则当且仅当$\rho(B)<1$时，A是M-矩阵。其中$B=D^{-1}C$，$A=D-C$，$D=\mathrm{diag}(a_{11}, a_{22}, \cdots, a_{NN})$，$\rho(B)$为$B$的谱半径。

7.3.27 定理 如果$n \times n$的实矩阵A是正定的且

$a_{ij} \leq 0 \quad (i \neq j, \quad i,j=1,2,\cdots,N)$

则A是M-矩阵。

7.3.28 定理 设A是M-矩阵，$\widetilde{A}$是把A中某些非对角线上的非零元素置为零后得到的矩阵，则$\widetilde{A}$也是非奇异的M-矩阵。

7.3.29 定义 设A为$n \times n$的非奇异阵，$A=M-N$而且$M^{-1} \geq 0$，$N \geq 0$成立，则称$A=M-N$为A的一个**正则分解**(Regular Splitting)。

7.3.30 推论 设A是非奇异的M-矩阵，而A_1是把A中某些非对角线的非零元素置为零后得到的矩阵，则$A=A_1-A_2$为一个正则分解。

7.3.31 例 M-矩阵

$$
A=\begin{pmatrix}
4 & -1 & & & -1 & & \\
-1 & 4 & -1 & & & -1 & \\
 & -1 & 4 & -1 & & & -1 \\
 & & \ddots & \ddots & \ddots & & \\
-1 & & & \ddots & \ddots & \ddots & \\
 & -1 & & & \ddots & \ddots & -1 \\
 & & -1 & & & -1 & 4
\end{pmatrix}
$$

分解为

$$
A_1=\begin{pmatrix}
4 & -1 & & & \\
-1 & 4 & -1 & & \\
 & -1 & 4 & -1 & \\
 & & \ddots & \ddots & \ddots \\
 & & & \ddots & \ddots & -1 \\
 & & & & -1 & 4
\end{pmatrix},\quad
A_2=\begin{pmatrix}
 & & 1 & & \\
 & & & 1 & \\
 & & & & \ddots \\
 & & & & & 1 \\
1 & & & & & \\
 & 1 & & & & \\
 & & \ddots & & & \\
 & & & 1 & &
\end{pmatrix}
$$

则$A=A_1-A_2$是一个正则分解.

7.4

7.4 迭代法的收敛性

用迭代法求解线性代数方程组

$Ax=f$

设 $A=I_n-B$，则有关于迭代法收敛和收敛速度的基本定理。

7.4.1 定理 设有方程组

$x=Bx+f$

对于任意初始向量$x^{(0)}$及任意f，解此方程组的迭代法为

7.4.2 $x^{(k+1)}=Bx^{(k)}+f \qquad (k=0,\ 1,\ 2,\ \cdots)$

它收敛的充分必要条件是迭代矩阵B的谱半径

7.4.3 $\rho(B)<1$

这个定理是迭代法收敛的基本定理。

7.4.4 定理 设有方程组

$$x = Bx + f$$

对于任意初始向量$x^{(0)}$和任意f，解此方程组的迭代公式为

$$x^{(k+1)} = Bx^{(k)} + f$$

若迭代矩阵的某一范数 $\|B\| < 1$，则

1° 迭代法收敛；

2° 事后误差估计

$$\|x^{(k)} - x^*\| \leqslant \frac{\|B\|}{1 - \|B\|} \|x^{(k)} - x^{(k-1)}\|$$

3° 事后误差估计

$$\|x^{(k)} - x^*\| \leqslant \frac{\|B\|^k}{1 - \|B\|} \|x^{(1)} - x^{(0)}\|$$

这个定理是迭代法收敛的充分条件，当 $\|B\| < 1$时，迭代收敛，而当 $\|B\| > 1$ 时，迭代不一定不收敛。x^*是方程组的精确解。

7.4.5 例 设方程组

$$x = Bx + f$$

其中

$$B = \begin{bmatrix} 0.9 & 0 \\ 0.3 & 0.8 \end{bmatrix}, \qquad f = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$$

则 $\|B\|_\infty = 1.1$，$\|B\|_1 = 1.2$，$\|B\|_2 = 1.021$，$\|B\|_F = \sqrt{1.54}$

显然B的这些范数都大于1，但B的特征值 $\lambda_1 = 0.9$，$\lambda_2 = 0.8$ 两个特征值均小于 1，由定理（7.4.1），迭代是收敛的，因此，$\|B\| < 1$ 是迭代法收敛的充分条件而不是充分必要条件。

7.4.6 定义/平均收敛速度 设$A \in R^{n\times n}$，$B \in R^{n\times n}$，如果对某一整数k，有 $\|A^k\| < 1$，则称

$$R(A^k) \equiv -\log \|A^k\|^{1/k} = \frac{-\log\|A^k\|}{k}$$

是矩阵A迭代k次的**平均收敛速度**(Average Rate of Convergence)。如果$R(A^k)<R(B^k)$，则B迭代k次比A迭代k次收敛快。

令$\varepsilon^{(k)}=x^{(k)}-x^*$，$x^*$是精确值，则

$$\varepsilon^{(k)}=B^k\varepsilon^{(0)}$$

由矩阵范数和向量范数的相容条件(定理7.2.11)

$$\|\varepsilon^{(k)}\| \leqslant \|B^k\| \cdot \|\varepsilon^{(0)}\|$$

则有

$$\frac{\|\varepsilon^{(k)}\|}{\|\varepsilon^{(0)}\|}\leqslant \|B^k\|$$

令 $\sigma=\left(\frac{\|\varepsilon^{(k)}\|}{\|\varepsilon^{(0)}\|}\right)^{1/k}$ 表示误差的平均衰减因子，因此，有

7.4.7 $\sigma\leqslant(\|B^k\|)^{1/k}=e^{-R(B^k)}$

这说明$R(B^k)$是误差平均衰减因子的指数衰减率。

令$N_k=(R(A^k))^{-1}$，由(7.4.7)知$\sigma^{N_k}\leqslant\frac{1}{e}$，因此，$N_k$是将初始误差减小$e$倍所需要的迭代次数。平均收敛速度$R(A^k)$愈大，迭代次数$N_k$愈小。

7.4.8 **定义/渐近收敛速度** 称$R_\infty(B)=-\log\rho(B)$为迭代法的**渐近收敛速度**(Asymptotic Rate of Convergence)。

考察误差向量$\varepsilon^{(k)}=x^{(k)}-x^*=B^k\varepsilon^{(0)}$，设$B$有$n$个线性无关的特征向量$u_1, u_2, \cdots, u_n$，相应的特征值为$\lambda_1, \lambda_2, \cdots, \lambda_n$。由

$$\varepsilon^{(0)}=\sum_{i=1}^{n}a_iu_i$$

$$\varepsilon^{(k)}=B^k\varepsilon^{(0)}=\sum_{i=1}^{n}a_iB^ku_i=\sum_{i=1}^{n}a_i\lambda_i^ku_i$$

可以看出当$\rho(B)<1$愈小时，$\lambda_i^k\to0(i=1,2,\cdots,n)(k\to\infty)$愈快，即$\varepsilon^{(k)}\to0$愈快，故$\rho(B)$刻划了迭代法的收敛快慢。

用k表示使$[\rho(B)]^k \leqslant 10^{-s}$所需要的迭代次数，取对数得

$$k \geqslant \frac{s\log 10}{-\log\rho(B)}$$

由此看出，$\rho(B)<1$愈小，$-\log\rho(B)$愈大，k愈小。

令$N_\infty=(R_\infty(B))^{-1}$，它是将$e^{(0)}$减小$e$倍所需的迭代次数。

7.4.9 **定理** 设$B\in R^{n\times n}$，如果k充分大时有$\|B^k\|<1$，则

$$\lim_{k\to\infty}R(B^k)=-\log\rho(B)\equiv R_\infty(B)$$

7.4.10 **推论** 设$B\in \boldsymbol{R}^{n\times n}$，对某个$k$，当$\|B^k\|<1$时，有

$$R_\infty(B)\geqslant R(B^k)$$

一般情况下使用$R_\infty(B)$，用它来刻划一个迭代矩阵的收敛速度，既简单又实际，但决不能乱用。比如，有一矩阵

$B=\begin{bmatrix} \alpha & 4 \\ 0 & \alpha \end{bmatrix}$，如果$\alpha=0.99$，则$R_\infty(B)=0.01005$，其倒数是$N_\infty=99.5$，这说明将任意初始误差减小$e$倍，大约需要迭代100次，但是，当$k<805$时，$\|B^k\|\geqslant 1$，为了使$\|B^k\| \leqslant \frac{1}{e}$，则需要$k\geqslant 918$。

7.5

7.5 Jacobi迭代法和RF法

Jacobi（**雅可比**）**迭代法**又称简单迭代法.

设有方程组

7.5.1 $$\sum_{j=1}^{n} a_{ij}x_j=b_i \quad (i=1,2,\cdots,n)$$

简记为

$$\boldsymbol{Ax}=\boldsymbol{b}$$

A为非奇异阵且$a_{ii}\neq 0(i=1,2,\cdots,n)$，则方程组（7.5.1）有唯一解，且可改写成

$$x_i = -\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j + \frac{b_i}{a_{ii}} \quad (i=1,2,\cdots,n)$$

7.5.2 Jacobi迭代公式

$$x_i^{(k+1)} = -\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}} \quad (i=1,2,\cdots,n)$$

7.5.3 Jacobi 迭代法的计算步骤如下：

1° 任意选取一组初始近似值$x_1^{(0)}$，$x_2^{(0)}$，$\cdots, x_n^{(0)}$作为方程的第0次近似解。

2° 依次使$k=0$，1，…，用公式

$$x_i^{(k+1)} = -\sum_{\substack{j=1\\ j\neq i}}^{n} \frac{a_{ij}}{a_{ii}} x_j^{(k)} + \frac{b_i}{a_{ii}} \qquad (i=1,2,\cdots n)$$

求出方程的第k次近似值，直至满足

$$\frac{\| x^{(k+1)} - x^{(k)} \|_\infty}{\| x^{(k+1)} \|_\infty} < \varepsilon$$

为止，式中ε为预先给定的允许相对误差。

7.5.4 **例** 用Jacobi迭代法解下列方程组

$$\begin{cases} 10x_1 - x_2 + 2x_3 = 6 \\ -x_1 + 11x_2 - x_3 + 3x_4 = 25 \\ 2x_1 - x_2 + 10x_3 - x_4 = -11 \\ 3x_2 - x_3 + 8x_4 = 15 \end{cases}$$

解 将方程组写成Jacobi迭代公式

7.5.5
$$\begin{cases} x_1^{(k+1)} = \frac{1}{10}x_2^{(k)} - \frac{1}{5}x_3^{(k)} + \frac{3}{5} \\ x_2^{(k+1)} = \frac{1}{11}x_1^{(k)} + \frac{1}{11}x_3^{(k)} - \frac{3}{11}x_4^{(k)} + \frac{25}{11} \\ x_3^{(k+1)} = -\frac{1}{5}x_1^{(k)} + \frac{1}{10}x_2^{(k)} + \frac{1}{10}x_4^{(k)} - \frac{11}{10} \\ x_4^{(k+1)} = -\frac{3}{8}x_2^{(k)} + \frac{1}{8}x_3^{(k)} + \frac{15}{8} \end{cases}$$

取初始向量$x^{(0)}=(0, 0, 0, 0)^T$，按上式进行迭代，结果列于表(7.5.6)，迭代进行到

$$\frac{\| x^{(k+1)} - x^{(k)} \|_\infty}{\| x^{(k+1)} \|_\infty} < 10^{-3}$$

为止。

7.5.6 表 Jacobi法计算结果

k	$x_1^{(k)}$	$x_2^{(k)}$	$x_3^{(k)}$	$x_4^{(k)}$
0	0.0000	0.0000	0.0000	0.0000
1	0.6000	2.2727	−1.1000	1.8750
2	1.0473	1.7159	−0.8052	0.8852
3	0.9326	2.0533	−1.0493	1.1309
4	1.0152	1.9537	−0.9681	0.9739
5	0.9890	2.0114	−1.0103	1.0214
6	1.0032	1.9922	−0.9945	0.9944
7	0.9981	2.0023	−1.0020	1.0036
8	1.0006	1.9987	−0.9990	0.9989
9	0.9997	2.0004	−1.0004	1.0006
10	1.0001	1.9998	−0.9998	0.9998

事实上

$$\frac{\| x^{(10)} - x^{(9)} \|_\infty}{\| x^{(10)} \|_\infty} = \frac{8.0\times 10^{-4}}{1.9998} < 10^{-3}$$

因此，迭代10次即满足精确度。

公式（7.5.1）中的矩阵A可以分裂为

7.5.7
$$A=\begin{pmatrix} a_{11} & & & \\ & a_{22} & & \\ & & \ddots & \\ & & & a_{nn} \end{pmatrix}-\begin{pmatrix} 0 & & & \\ -a_{21} & \ddots & & \\ \vdots & \ddots & \ddots & \\ -a_{n1} & \cdots & -a_{n,n-1} & 0 \end{pmatrix}$$

$$-\begin{pmatrix} 0 & -a_{12} & \cdots & -a_{1n} \\ & \ddots & \ddots & \vdots \\ & & \ddots & -a_{n-1,n} \\ & & & 0 \end{pmatrix}$$

$$=D-L-U$$

其中D为对角矩阵，L为严格下三角矩阵，U为严格上三角阵。用矩阵D，L，U来表示Jacobi迭代公式(7.5.3)，可以得到

7.5.8 Jacobi迭代法的矩阵公式

1° $\boldsymbol{x}^{(0)}=(x_1^{(0)},\ x_2^{(0)},\ \cdots x_n^{(0)})$为任意给定的初始向量。

2° $\boldsymbol{x}^{(k+1)}=B_J\boldsymbol{x}^{(k)}+\boldsymbol{f}_J\ (k=0,\ 1,\ 2,\ \cdots)$

其中$B_J=D^{-1}(L+U)=I_n-D^{-1}A$，$\boldsymbol{f}_J=D^{-1}\boldsymbol{b}$，称$B_J$为Jacobi**迭代矩阵**(Iteration Matrix)。

Jacobi方法公式简单，每迭代一次只需计算一次矩阵和向量的乘积。编制程序简单，但对x需要二套存储单元，以便存储$\boldsymbol{x}^{(k)}$及$\boldsymbol{x}^{(k+1)}$。如果方程组确有解，为防止电子计算机计算时溢出，在编程序前应整理公式，使$a_{ii}\neq 0(i=1,2,\cdots,n)$，为了使收敛速度快，$a_{ii}$的绝对值应尽可能地大。

7.5.9 Jacobi迭代法的收敛性

1° Jacobi迭代法收敛的充分必要条件是Jacobi迭代矩阵B_J的谱半径$\rho(B_J)<1$。

2° 如果 B_J 的某一种范数 $\|B_J\|<1$，则Jacobi迭代法收敛.比如，当

$$\|B_J\|_\infty=\max_{1\leq i\leq n}\sum_{j=1}^{n}|(B_J)_{ij}|<1$$

或

$$\|B_J\|_1=\max_{1\leq j\leq n}\sum_{i=1}^{n}|(B_J)_{ij}|<1$$

或

$$\|B_J\|_F=\left(\sum_{i=1}^{n}\sum_{j=1}^{n}(B_J)^2_{ij}\right)^{1/2}<1$$

时，Jacobi迭代法收敛.

3° 如果 $A\boldsymbol{x}=\boldsymbol{b}$ 中的 A 是严格对角占优阵(定义7.3.3,)则Jacobi迭代法收敛.

4° 如果 A 对称正定且有性质'$\boldsymbol{A}$'，或者 A 不可约且弱对角占优，则 $\rho(B_J)<1$，Jacobi迭代法收敛.

Jacobi迭代法第 k 次近似解的误差按定理(7.4.4)中的2°或3°估算.

当(7.5.7)式中 $D=I_n$ 即 $A=I_n-L-U$ 时，(7.5.8)式变成

7.5.10 $$\begin{aligned}\boldsymbol{x}^{(k+1)}&=(L+U)\boldsymbol{x}^{(k)}+\boldsymbol{b}\\&=(I_n-A)\boldsymbol{x}^{(k)}+\boldsymbol{b}\\&=B_{RF}\boldsymbol{x}^{(k)}+\boldsymbol{b}\end{aligned}$$

其中 $B_{RF}=I_n-A$. 称(7.5.10)为**RF迭代公式**,称 B_{RF} 为**RF迭代矩阵**.

RF法是Richardson(李查逊)方法的一种变型,B_{RF} 的特征值 $\lambda(B_{RF})$ 与 A 的特征值 $\lambda(A)$ 之间的关系是：$\lambda(B_{RF})=1-\lambda(A)$。因此，$\rho(I_n-A)=\max(|1-\lambda_{\min}(A)|,|1-\lambda_{\max}(A)|)$,

其中$\lambda_{\min}(A)$和$\lambda_{\max}(A)$分别是矩阵A的最小特征值和最大特征值。由此可知，RF迭代法收敛的充分必要条件是

7.5.11 $\lambda_{\max}(A)<2$

7.6

7.6 Gauss-Seidel迭代法

Gauss-Seidel（高斯一赛得尔）迭代法简称为 G-S迭代法，求解方程组（7.5.1）的迭代公式为

7.6.1
$$x_i^{(k+1)}=-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(k+1)}-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{(k)}+\frac{b_i}{a_{ii}}$$
$$(i=1,2,\cdots,n)$$

G-S迭代法的计算步骤同 Jacobi 迭代法，仅是第 2 步中的计算公式改用(7.6.1)。

7.6.2 **例** 用G-S迭代法解例（7.5.4）中的方程组。

解 将方程组写成G-S迭代公式

7.6.3
$$\begin{cases} x_1^{(k+1)}= \frac{1}{10}x_2^{(k)}-\frac{1}{5}x_3^{(k)}+\frac{3}{5} \\ x_2^{(k+1)}=\frac{1}{11}x_1^{(k+1)}+\frac{1}{11}x_3^{(k)}-\frac{3}{11}x_4^{(k)}+\frac{25}{11} \\ x_3^{(k+1)}=-\frac{1}{5}x_1^{(k+1)}+\frac{1}{10}x_2^{(k+1)}+\frac{1}{10}x_4^{(k)}-\frac{11}{10} \\ x_4^{(k+1)}=-\frac{3}{8}x_2^{(k+1)}+\frac{1}{8}x_3^{(k+1)}+\frac{15}{8} \end{cases}$$

和Jacobi迭代法一样取初始向量$x^{(0)}=(0,0,0,0)^T$，按上式进行迭代，迭代进行到

$$\frac{\|x^{(k+1)}-x^{(k)}\|_\infty}{\|x^{(k+1)}\|_\infty}<10^{-3}$$

为止，结果列于表(7.6.4)。

7.6.4 表 G-S迭代法计算结果

k	0	1	2	3	4	5
$x_1^{(k)}$	0.0000	0.6000	1.030	1.0065	1.0009	1.0001
$x_2^{(k)}$	0.0000	2.3272	2.037	2.0036	2.0003	2.0000
$x_3^{(k)}$	0.0000	−0.9873	−1.014	−1.0025	−1.0003	−1.0000
$x_4^{(k)}$	0.0000	0.8789	0.9844	0.9983	0.9999	1.0000

事实上，$\dfrac{\|x^{(5)}-x^{(4)}\|_\infty}{\|x^{(5)}\|_\infty}=\dfrac{0.0008}{2.000}=4\times10^{-4}<10^{-3}$，因此，迭代五次满足精确度。

与（7.6.3）式等价的矩阵表达式为

7.6.5 $x^{(k+1)}=(D-L)^{-1}Ux^{(k)}+(D-L)^{-1}b$

其中D、L、U同(7.5.7)。于是，可得到

7.6.6 G-S迭代法的矩阵公式

1° $x^{(0)}=(x_1^{(0)},x_2^{(0)},\cdots,x_n^{(0)})$为任意给定的初始向量。

2° $x^{(k+1)}=B_{GS}x^{(k)}+f_{GS}$

其中 $B_{GS}=(D-L)^{-1}U$，$f_{GS}=(D-L)^{-1}b$，称 B_{GS}为G-S迭代矩阵。

和 Jacobi 迭代法相比，G-S迭代法有二个明显的优点：

1° 只需要一套存贮单元存放近似解，比Jacobi 方法省一套单元。

2° 在第$k+1$次迭代中，最新计算出来的分量如 $x_1^{(k+1)}$，$x_2^{(k+1)},\cdots x_i^{(k+1)}$，在计算第$i+1$个分量时加以利用。

从例(7.6.2)可知，达到同样的精度10^{-3}，G－S迭代法比Jacobi迭代法所需要的迭代次数少，即G-S 法收敛快．但这个结论只在一定的条件下成立。对有的方程组，Jacobi 方法收敛，而G－S方法却是发散的。

7.6.7 G－S迭代法的收敛性

1° 收敛的充分必要条件是G－S迭代矩阵的谱半径$\rho(B_{GS})<1$。

2° 如果 $Ax=b$ 中的矩阵 A 是对称正定的（定义7.3.10），则G－S迭代法收敛。

3° 如果A是严格对角占优的（定义7.3.3），则G－S迭代法收敛。

4° 如果G－S法的迭代矩阵B_{GS}的某一种范数 $\|B_{GS}\|<1$，则G－S迭代法收敛。

G－S迭代法第 k 次近似解的误差按定理（7.4.4）中的2°或3°估算。

7.7

7.7 SOR迭代法

SOR是Successive Over Relaxation（逐次超松弛）的缩写.SOR迭代法是解大型稀疏矩阵方程组的有效方法之一。求解方程组（7.5.1）的SOR迭代公式是

7.7.1
$$x_i^{(k+1)}=(1-\omega)\,x_i^{(k)}+\omega\left(\frac{b_i}{a_{ii}}-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(k+1)}\right.$$
$$\left.-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}\;x_j^{(k)}\right)\qquad(i=1,2,\cdots n)$$

称ω为**松弛参数**或**松弛因子**(Relaxation Factor)，称$0<\omega<1$的迭代过程（7.7.1）为**低松弛方法**（Under-relaxation Method）。对于一些方程组，用G-S迭代法得不到收敛解或不收敛，但用低松弛方法却是收敛的。称$\omega>1$的迭代过程(7.7.1)为**超松弛方法**（Over-relaxation Method），此法可以加速G-S迭代方法的收敛。$\omega=1$的迭代过程（7.7.1）就是G-S迭代公式.

SOR迭代法的计算步骤同Jacobi迭代法，仅是第2步中的计算公式改用(7.7.1)。

7.7.2 例 用SOR法和G-S法解方程组

$$\begin{cases} 4x_1+3x_2 \qquad = 24 \\ 3x_1+4x_2-x_3 = 30 \\ \qquad -x_2+4x_3 = -24 \end{cases}$$

方程组有精确解$x^*=(3,4,-5)^T$.

解 由SOR的迭代公式（7.7.1）得

$$\begin{cases} x_1^{(k+1)}=x_1^{(k)}+\dfrac{\omega}{4}(24-4x_1^{(k)}-3x_2^{(k)}) \\ x_2^{(k+1)}=x_2^{(k)}+\dfrac{\omega}{4}(30-3x_1^{(k+1)}-4x_2^{(k)}+x_3^{(k)}) \\ x_3^{(k+1)}=x_3^{(k)}+\dfrac{\omega}{4}(-24+x_2^{(k+1)}-4x_3^{(k)}) \end{cases}$$

$$(k=0,1,2,\cdots)$$

当$\omega=1$时，即G－S迭代公式，有

$$\begin{cases} x_1^{(k+1)}=0.75x_2^{(k)}+6 \\ x_2^{(k+1)}=-0.75x_1^{(k+1)}+0.25x_3^{(k)}+7.5 \\ x_3^{(k+1)}=0.25x_2^{(k+1)}-6 \end{cases}$$

$$(k=0,1,2,\cdots)$$

$\omega=1.25$的SOR迭代公式为

$$\begin{cases} x_1^{(k+1)}=-0.25x_1^{(k)}-0.9375x_2^{(k)}+7.5 \\ x_2^{(k+1)}=-0.9375x_1^{(k+1)}-0.25x_2^{(k)} \\ \qquad\qquad +0.3125x_3^{(k)}+9.375 \\ x_3^{(k+1)}=0.3125x_2^{(k+1)}-0.25x_3^{(k)}-7.5 \end{cases}$$

$$(k=0,1,2,\cdots)$$

上面两组方程都取$x^{(0)}=(1,1,1)^T$,迭代进行到$\|x^{(k+1)}-x^{(k)}\|_\infty<\frac{1}{2}10^{-7}$.G-S计算结果列于表（7.7.3）;$\omega=1.25$, SOR计算结果列于表(7.7.4). 每种方法列 7 次迭代值.

7.7.3 表 G－S计算结果

k	0	1	2	3
$x_1^{(k)}$	1	5.250000	3.1406250	3.0878906
$x_2^{(k)}$	1	3.812500	3.8828125	3.9267578
$x_3^{(k)}$	1	－5.046875	－5.0292969	－5.0183105
k	4	5	6	7
$x_1^{(k)}$	3.0549316	3.0343323	3.0214577	3.0134110
$x_2^{(k)}$	3.9542236	3.9713898	3.9821186	3.9888241
$x_3^{(k)}$	－5.0114441	－5.0071526	－5.0044703	－5.0027940

7.7.4 表 $\omega=1.25$，SOR计算结果

k	0	1	2	3
$x_1^{(k)}$	1	6.312500	2.6223145	3.1333027
$x_2^{(k)}$	1	3.5195313	3.9585266	4.0102646
$x_3^{(k)}$	1	－6.6501465	－4.6004238	－5.0966863
k	4	5	6	7
$x_1^{(k)}$	2.9570512	3.0037211	2.9963276	3.0000498
$x_2^{(k)}$	4.0074838	4.0029250	4.0009262	4.0002586
$x_3^{(k)}$	－4.9734897	－5.0057135	－4.9982822	－5.0003486

达到相同精度$\frac{1}{2}\times10^{-7}$，则G－S迭代法需要迭代34次。

而SOR法（$\omega=1.25$）只需要迭代14次．由本例可以看出，SOR法的收敛速度比G－S法快．

将SOR的迭代公式（7.7.1）写成矩阵形式

$$Dx^{(k+1)}=(1-\omega)Dx^{(k)}+\omega(b+Lx^{(k+1)}+Ux^{(k)})$$

其中矩阵D,L,U是由A分裂来的，同(7.5.7)．于是，得到求解$Ax=b$的SOR迭代法的矩阵公式：

7.7.5 $\begin{cases} x^{(0)} \text{为任意给定的初始向量} \\ x^{(k+1)}=B_{SOR}x^{(k)}+f_{SOR} \end{cases}$

其中

$$B_{SOR}=(D-\omega L)^{-1}[(1-\omega)D+\omega U],f_{SOR}=\omega(D-\omega L)^{-1}b$$

称B_{SOR}为SOR迭代矩阵．

7.7.6 SOR迭代法的收敛性

1° 收敛的充分必要条件是SOR迭代矩阵的谱半径$\rho(B_{SOR})<1$。

2° 收敛的必要条件是松弛因子ω应满足条件$0<\omega<2$．

3° 如果$Ax=b$中的A是对称正定的（定义 7.3.10），且$0<\omega<2$，则SOR法收敛。

4° 如果B_{SOR}的某一种范数$\|B_{SOR}\|<1$，则SOR法收敛。

SOR迭代法第k次近似解的误差按定理（7.4.4）中的2°或3°估算．

7.8

7.8 松弛因子的选取

使SOR迭代法的渐近收敛速度$R_\infty(B_{SOR})$最快的松弛因子通常称为**最优松弛因子**(Optimum Rrelaxation Ractor)，用ω_{opt}记之．其中B_{SOR}为SOR迭代矩阵(7.7.5)．

7.8.1 最优松弛因子ω_{opt}的理论计算公式

对于一般矩阵(即使是对称正定矩阵)，目前尚无确定

ω_{opt}的理论结果，仅对某些特殊类型的矩阵，例如，对所谓有性质'A'（定义7.3.18）的矩阵，有确定ω_{opt}的理论公式。形如（7.3.16）的D-型分块三对角矩阵是具有性质A和相容次序（定义7.3.24）矩阵的一种特殊情况，由于实践中最常遇到对称正定矩阵（定义7.3.10），故有对称正定的D-型分块三对角阵的ω_{opt}的理论计算公式。

7.8.1 定理 假定方程组$Ax=b$的系数矩阵A为对称正定且具有（7.3.16）的形式，按（7.5.7）分裂$A=D-L-U$，其中D为对角矩阵，L和$U=L^T$分别为严格下三角形矩阵和严格上三角形矩阵，则$\rho(B_{GS})=[\rho(B_J)]^2$，且SOR方法的最优松弛因子$\omega_{opt}$为

7.8.2 $$\omega_{opt}=\frac{2}{1+\sqrt{1-[\rho(B_J)]^2}}$$

其中$\rho(B_J)$是 Jacobi 迭代矩阵$B_J=D^{-1}(L+U)$的谱半径，$\rho(B_{GS})$是G-S 迭代矩阵 $B_{GS}=(D-L)^{-1}U$ 的谱半径（定义7.2.17）。

由于三对角矩阵是形如（7.3.16）矩阵的特例，故定理（7.8.1）对于对称正定的三对角阵亦成立。

例 试确定SOR法解方程组

$$\begin{cases} 4x_1+3x_2 \qquad = 24 \\ 3x_1+4x_2-x_3= 30 \\ \qquad -x_2+4x_3=-24 \end{cases}$$

的最优松弛因子ω_{opt}。

解 已知

$$A=\begin{pmatrix} 4 & 3 & 0 \\ 3 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix}$$

是对称正定三对角阵，可以应用定理(7.8.1)，首先计算$\rho(B_J)$。由于$B_J=D^{-1}(L+U)$，而

$$D^{-1}(L+U)=\begin{pmatrix}\frac{1}{4} & & \\ & \frac{1}{4} & \\ & & \frac{1}{4}\end{pmatrix}\begin{pmatrix}0 & -3 & 0\\ -3 & 0 & 1\\ 0 & 1 & 0\end{pmatrix}=\begin{pmatrix}0 & -0.75 & 0\\ -0.75 & 0 & 0.25\\ 0 & 0.25 & 0\end{pmatrix}$$

于是$\det(B_J-\lambda I)=-\lambda(\lambda^2-0.625)=0$，则有

$\rho(B_J)=\sqrt{0.625}$

故由（7.8.2）式有

$$\omega_{opt}=\frac{2}{1+\sqrt{1-0.625}}\approx 1.24$$

因此，例（7.7.2）中$\omega=1.25$比较接近ω_{opt}。

7.8.3 定理 若矩阵$A=I-L-L^T$为对称正定矩阵（对角线元为1，下三角部分为$-L$），A的特征值为$\lambda_1\geqslant\cdots\geqslant\lambda_n>0$，矩阵$H=-L^T-L$之谱半径$\rho(H)=\sigma$，则可按如下公式粗略地确定$\omega_{opt}$。

$$\omega_{opt}\approx\omega^*=\begin{cases}\dfrac{2}{1+\sqrt{\lambda_1\cdot\lambda_n-\sigma^2}}, & 当\lambda_n\cdot(\lambda_1-\lambda_n)>2\sigma^2\\[2ex] \dfrac{2}{1+\sqrt{\lambda_n^2+\sigma^2}}, & 当\lambda_n\cdot(\lambda_1-\lambda_n)\leqslant 2\sigma^2\end{cases}$$

实际计算时，λ_1和σ可用其上界，λ_n可用其下界(大于零)。

7.8.4 定理 假定矩阵$A=D-L-U$的对角线部分D为非奇异，且矩阵$B_J=D^{-1}(L+U)$满足如下条件：

1° B_J的所有元素均非负，即$B_J\geqslant 0$；

2° B_J是不可约的（定义7.3.1），且$0<\rho(B_J)<1$；

3° B_J为对称矩阵，

则当SOR法中松弛因子取为

$\omega_{opt}=\dfrac{2}{1+\sqrt{1-\rho^2(B_J)}}$时，SOR迭代矩阵的谱半径$\rho(B_{SOR})$应满足如下关系式：

7.8.5 $\omega_{opt}-1\leqslant\rho(B_{SOR})<\sqrt{\omega_{opt}-1}$

当且仅当B为具有相容次序（定义7.3.24）和性质 A（定义7.3.18）的矩阵时，

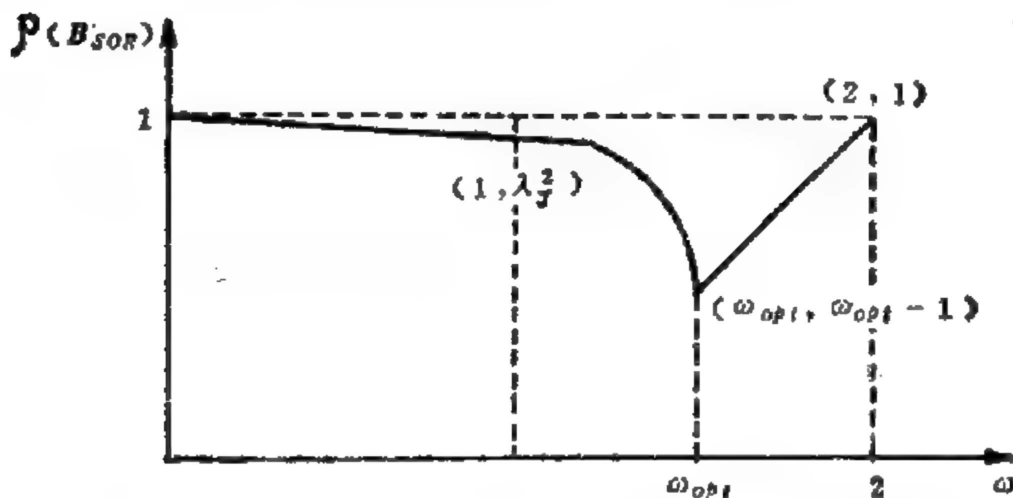
$$\rho(B_{SOR})=\omega_{opt}-1=\left(\frac{\rho(B_J)}{1+\sqrt{1-\rho^2(B_J)}}\right)^2<\rho^2(B_J)<\rho(B_J)$$

对于一般的松弛参数ω有

7.8.6 $$\rho(B_{SOR})=\begin{cases}\left(\dfrac{\omega\rho(B_J)+\sqrt{\omega^2\rho(B_J)^2-4(\omega-1)}}{2}\right)^2 & 0<\omega<\omega_{opt}\\ \omega-1 & \omega\geqslant\omega_{opt}\end{cases}$$

见图(7.8.7)。

7.8.6 图



由图（7.8.7）可知，当$\omega<\omega_{opt}$并逐渐增加趋近于ω_{opt}时，$\rho(B_{SOR})$曲线的切线趋近于铅垂线，但当$\omega>\omega_{opt}$并逐渐减小

趋近于ω_{opt}时，该切线方向不变，其斜率恒为1.

7.8.2 ω_{opt}的试算

对于大多数矩阵，目前还没有计算ω_{opt}的理论公式，即使有理论公式（7.8.2）和（7.8.4），其中$\rho(B_J)$等一些参数也难于预先确定，故在实际计算中只能通过试算确定ω_{opt}.

7.8.8 最简单的试算确定办法

取不同的松弛因子$\omega_1, \omega_2, \cdots$，从同一个初始向量$x^{(0)}$出发，用SOR迭代公式（7.7.1）进行迭代，迭代次数同为k（迭代次数不应太少），然后比较用ω_1，ω_2，…算出的剩余$r_{\omega_1}=b-Ax_{\omega_1}^{(k)}$，$r_{\omega_2}=b-Ax_{\omega_2}^{(k)}$，…或误差$\|x_{\omega_1}^{(k)}-x_{\omega_1}^{(k-1)}\|$，$\|x_{\omega_2}^{(k)}-x_{\omega_2}^{(k-1)}\|$，…. 其中$x_{\omega_i}^{(k)}$是用$\omega_i$迭代$k$次得到的$Ax=b$的近似解. 选取达到$\min\limits_i\|r_{\omega_i}\|$或$\min\limits_i\|x_{\omega_i}^{(k)}-x_{\omega_i}^{(k-1)}\|$（即使得剩余或误差的范数(或模)最小）的松弛因子作为ω_{opt}的近似值. 这个方法简单而有效，特别当使用者需要多次求解具有相同系数矩阵的方程组时更是如此. 另外，如果使用者对于所求解的方程组有较深入了解或积累了一定经验，常常可以事先定出一个包含ω_{opt}的不大的区间〔ω_a，ω_b〕，这样可以大大减少试算次数，较快地确定ω_{opt}的近似值. 也可以采用优选法的原则从区间〔ω_a，ω_b〕中选取进行试算的ω值，以便更快地找到ω_{opt}的较好的近似值.

7.8.9 选取ω_{opt}的自适应方法

1° 选一个ω使$\omega<\omega_{opt}$，比如令$\omega=1$；

2° 作n次SOR迭代，用连续二次伪余量之比

$$\alpha^{(k)}=\frac{\|\Delta x^{(k)}\|_2}{\|\Delta x^{(k-1)}\|_2}=\frac{\left(\sum\limits_{j=1}^{n}|x_j^{(k+1)}-x_j^{(k)}|^2\right)^{1/2}}{\left(\sum\limits_{j=1}^{n}|x_j^{(k)}-x_j^{(k-1)}|^2\right)^{1/2}}$$

作为$\rho(B_{SOR})$的估值，其中伪余量$\Delta x^{(k)}=x^{(k+1)}-x^{(k)}$。从公式（7.7.5）可知

$$\Delta x^{(k)}=B_{SOR}\Delta x^{(k-1)}$$

所以，当k相当大时，

$$\rho(B_{SOR})\approx\frac{\|\Delta x^{(k)}\|}{\|\Delta x^{(k-1)}\|}$$

3° 由$\alpha^{(k)}$和ω计算相应的$\rho(B_J)$

公式（7.8.6）的第一式等价于

$$\rho(B_{SOR})+\omega-1=\omega\rho(B_J)\sqrt{\rho(B_{SOR})}$$

从而有

$$\rho(B_J)\approx\frac{\alpha^{(k)}+\omega-1}{\omega\sqrt{\alpha^{(k)}}}=\beta^{(k)}$$

4° 由$\rho(B_J)$的估值$\beta^{(k)}$去计算改进的ω_{opt}

由（7.8.2）算出ω_{opt}的第一次近似值

$$\omega_{opt}^{(1)}=\frac{2}{1+\sqrt{1-\beta^{(k)2}}}$$

重复2°—4°可得一系列近似的$\omega_{opt}^{(1)}$，$\omega_{opt}^{(2)}$，，…，直到相邻两数之差不超过10^{-2}(或某个适当小的正数)为止，往下可利用这个松弛因子进行迭代直到结束。

7.8.10 另一个近似选取ω_{opt}的方法

令

$$\delta^{(k)}=\max_{i,j}|x_{ij}^{(k)}-x_{ij}^{(k-1)}|$$

由$\omega=1$开始迭代，直到下述条件成立

$$\lg\delta^{(k-2)}-\lg\delta^{(k-1)}\approx\lg\delta^{(k-1)}-\lg\delta^{(k)}=a$$

其中a是与k无关的常数。

然后，近似地选取最优松弛因子为

$$\omega_{opt}\approx\frac{2}{1+\sqrt{1-10^{-a}}}$$

7.9

7.9 SSOR迭代法

SSOR是 Symmetric Successive Over Relaxation（对称逐次超松弛）的缩写。SSOR方法是指如下的迭代过程：假设已知第k次迭代的近似值 $x^{(k)}=(x_1^{(k)},x_2^{(k)},\cdots x_n^{(k)})$，先按照自然次序($i=1,2,\cdots n$)用向前的SOR法逐点计算$x_i^{(k+\frac{1}{2})}$，即

7.9.1 $$x_i^{(k+\frac{1}{2})}=(1-\omega)x_i^{(k)}+\omega\left(\frac{b_i}{a_{ii}}-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(k+\frac{1}{2})}-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{(k)}\right)$$

$$(i=1,2,\cdots n)$$

然后按照相反的次序($i=n,\ n-1,\ \cdots 1$)用向后的SOR方法逐点计算$x_i^{(k+1)}$，即

7.9.2 $$x_i^{(k+1)}=(1-\omega)x_i^{(k+\frac{1}{2})}+\omega\left(\frac{b_i}{a_{ii}}-\sum_{j=1}^{i-1}\frac{a_{ij}}{a_{ii}}x_j^{(k+\frac{1}{2})}-\sum_{j=i+1}^{n}\frac{a_{ij}}{a_{ii}}x_j^{(k+1)}\right)$$

$$(i=n,n-1,\cdots 1)$$

(7.9.1) 和 (7.9.2) 中的松弛因子ω用相同的值，$0<\omega<2$。

将SSOR方法的迭代公式写成矩阵形式

7.9.3 $$\begin{cases}x^{(k+\frac{1}{2})}=B_{fSOR}x^{(k)}+f_{fSOR}\\ x^{(k+1)}=B_{BSOR}x^{(k+\frac{1}{2})}+f_{BSOR}\end{cases}$$

其中

7.9.4 $$\begin{cases}B_{fSOR}=(D-\omega L)^{-1}[\omega U+(1-\omega)D]\\ B_{BSOR}=(D-\omega U)^{-1}[\omega L+(1-\omega)D]\end{cases}$$

因此，SSOR迭代矩阵是

7.9.5 $B_{SSOR}=(D-\omega U)^{-1}[\omega L+(1-\omega)D](D-\omega L)^{-1}[\omega U+(1-\omega)D]$

以上各式中的D、L、U是由A分裂来的，同(7.5.7)。

松弛因子ω对SSOR迭代收敛速度的影响不像SOR法那样敏感，因此，它并不需要很精确的最优松弛因子ω_{opt}（见7.8）。

7.9.6 SSOR迭代方法的收敛性

1° 如果方程组$Ax=b$中的A是对称正定矩阵，且$0<\omega<2$，则B_{SSOR}的特征值$\lambda\in[0,1)$，即$\rho(B_{SSOR})<1$，所以SSOR迭代是收敛的。

2° 如果$\rho(D^{-1}LD^{-1}U)\leqslant\frac{1}{4}$，则SSOR迭代收敛。对于大多数广义Dirichlet（狄利希莱特）问题，只要网格节点以自然次序排列，则条件$\rho(D^{-1}LD^{-1}U)\leqslant\frac{1}{4}$满足。

7.9.7 松弛因子ω的选取

1° 如果$\rho(D^{-1}LD^{-1}U)\leqslant\frac{1}{4}$，已知相应的Jacobi迭代矩阵的按模最大特征值$\lambda_J$，则可以由

7.9.8 $\omega^*=\dfrac{2}{1+\sqrt{2(1-\lambda_J)}}$

计算得到较好的松弛因子ω^*，而且对于这个ω^*，能够证明

7.9.9 $\rho(B_{SSOR})\leqslant\dfrac{1-\sqrt{\dfrac{1-\lambda_J}{2}}}{1+\sqrt{\dfrac{1-\lambda_J}{2}}}$

2° 求得$\rho(D^{-1}LD^{-1}U)$的上界$\bar{\beta}\left(\geqslant\frac{1}{4}\right)$和$\lambda_J$的上界$\bar{\lambda}_J$，

则可得松弛因子ω^*。

7.9.10 $\omega^*=\dfrac{2}{1+\sqrt{1-2\bar{\lambda}_J+4\bar{\beta}}}$

7.10

7.10 最优外推法

通常使用的迭代公式均属**一阶线性定常迭代** (Linear Stationary Iteration)，即

7.10.1 $\boldsymbol{x}^{(k+1)}=B\boldsymbol{x}^{(k)}+M\boldsymbol{b}=B\boldsymbol{x}^{(k)}+\boldsymbol{f}$

其中$B+MA=I_n$，A是$A\boldsymbol{x}=\boldsymbol{b}$的系数矩阵.

应用到(7.10.1)式的**外推** (Extrapolation)方法是

7.10.2 $\boldsymbol{x}^{(k+1)}=\alpha(B\boldsymbol{x}^{(k)}+\boldsymbol{f})+(1-\alpha)\boldsymbol{x}^{(k)}$
$=B_\alpha\boldsymbol{x}^{(k)}+\alpha\boldsymbol{f}$

其中

7.10.3 $B_\alpha\equiv\alpha B+(1-\alpha)I_n$

α称为**外推因子** (Extrapolation Factor).

7.10.4 **定义** 如果有非奇异矩阵p使$p(I_n-B)p^{-1}$是对称正定矩阵（定义7.3.10），则称迭代方法（7.10.1）是**可对称化的** (Symmetrizable)，称P为**对称化矩阵** (Symmetrizable Matrix)。

7.10.5 **定理** 如果迭代方法（7.10.1）是可对称化的，则

1° B的特征值是实的；

2° B的最大特征值小于1，即$M(B)<1$。

如果迭代方法是可对称化的，则使$\rho(B_\alpha)$达到最小的**最优外推因子**α_{opt}是

7.10.6 $\alpha_{opt}=\dfrac{2}{2-M(B)-m(B)}$

其中$M(B)$和$m(B)$分别是B的最大特征值和最小特征值。

用α_{opt}代入(7.10.2)，有

7.10.7 $x^{(k+1)}=B_{\alpha_{opt}}x^{(k)}+\alpha_{opt}f$

用(7.10.7)进行迭代的方法称为**最优外推方法**。最优外推法迭代矩阵的谱半径小于1，即

7.10.8 $\rho(B_{\alpha_{opt}})=\dfrac{M(B)-m(B)}{2-M(B)-m(B)}<1$

因此最优外推方法(7.10.7)是收敛的。

由于Jacobi迭代法的最优外推方法是

7.10.9 $x^{(k+1)}=\alpha_{opt}(B_J x^{(k)}+f)+(1-\alpha_{opt})x^{(k)}$

所以，当矩阵A具有性质'$\boldsymbol{A}$'时，B_J(Jacobi迭代矩阵)的特征值满足

7.10.10 $m(B)=-M(B)$

将上式代入(7.10.6)式，得$\alpha_{opt}=1$，此时，最优Jacobi外推法变成没有外推的Jacobi方法。

7.11

7.11 块迭代法和隐式交替方向迭代法

7.11.1 分块迭代法的计算过程

在(7.5)到(7.10)中的几种基本迭代方法都是点迭代方法，分别称为**点Jacobi(雅可比)迭代法**、**点Gauss-Seidel(高斯-赛得尔)迭代法(简称为点G-S迭代法)**、**点SOR(逐次超松弛)迭代法**以及**点最优外推法**。它们的特点是每执行一个迭代公式只修正一个未知量，逐个修正直到求得x的全部分量，然后再重复此过程。

所谓**块迭代**(Partition Iteration)是把要求解的方程组和未知量分成组。迭代时，从每个方程组中同时确定相应的一组新的近似分量而不是一个量，逐组进行计算，直到求得全部新的近似分量为止，然后重复此过程。

从矩阵形式看，首先把方程组

$Ax=b$

中的矩阵A、未知向量x和右端项b进行分块如下

7.11.1
$$\begin{pmatrix} A_{11} & A_{12} & \cdots & A_{1M} \\ A_{21} & A_{22} & \cdots & A_{2M} \\ \vdots & & & \\ A_{M1} & A_{M2} & \cdots & A_{MM} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_M \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ \vdots \\ b_M \end{pmatrix}$$

其中A_{II}为n_I阶方阵，x_I和b_I为 n_I 维向量，A_{IJ}为$n_I\times n_J$矩阵。分块的原则是使迭代公式有较高的收敛速度。然后，将A_{IJ}、x_I和 b_I 看作元素， 仿照（7.5）到（7.10）的点迭代公式可得到相应的块迭代公式。比如，块 Jacobi迭代公式

7.11.2
$$\begin{cases} A_{11}x_1^{(k+1)} = \qquad -A_{12}x_2^{(k)} - \cdots - A_{1M}x_M^{(k)} + b_1 \\ A_{22}x_2^{(k+1)} = -A_{21}x_1^{(k)} \qquad -A_{23}x_3^{(k)} - \cdots \\ \qquad -A_{2M}x_M^{(k)} + b_2 \\ \qquad \cdots \\ A_{MM}x_M^{(k+1)} = -A_{M1}x_1^{(k)} - A_{M2}x_2^{(k)} - \cdots \\ \qquad -A_{M,M-1}x_{M-1}^{(k)} + b_M \end{cases}$$

假设已知$x_I^{(k)}(I=1,2,\cdots M)$，将它代入上列各方程的右端，则每个方程的右端项也已知，令其为$y_I^{(k)}(I=1,2,\cdots M)$，上列方程可写成

7.11.3 $A_{II}x_I^{(k+1)}=y_I^{(k)} \quad (I=1,2,\cdots M)$

块迭代的计算过程是先求解第一个低阶方程组

$A_{11}x_1^{(k+1)}=y_1^{(k)}$

由于方程组的阶数低，一般用直接方法如Gauss消去法(6.2)求解。通常 A_{II} 是三对角阵或带 型矩阵，这时可采用追赶法或带型矩阵消去法（6.6）求解。当矩阵 A 是对称正定时，A_{II}也应是对称正定的，可采用平方 根法。除了用直接方法之外也可用迭代法，在解出n_1个未 知量x_1后，从第二个方程组

$A_{22}x_2^{(k+1)}=y_2^{(k)}$

解出n_2个未知量$x_2^{(k+1)}$等等，这种逐块进行迭代的方法称为**块Jacobi迭代法**．在解二维椭圆型方程边值问题时，常常把一条或几条网格线上的所有未知数分在一个组内，然后逐线进行迭代，故又称为**线迭代方法**（Line Iteration Method）．

7.11.2 块（或线）迭代公式

7.11.4 线Jacobi迭代公式

$$A_{ii}x_i^{(k+1)}=-\sum_{\substack{j=1\\j\neq i}}^{M}A_{ij}x_j^{(k)}+b_i \qquad (i=1,2,\cdots M)$$

7.11.5 线G-S迭代公式

$$A_{ii}x_i^{(k+1)}=-\sum_{j=1}^{i-1}A_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{M}A_{ij}x_j^{(k)}+b_i$$

$$(i=1,2,\cdots M)$$

7.11.6 线SOR迭代公式

$$A_{ii}x_i^{(k+1)}=(1-\omega)x_i^{(k)}+\omega\Big(b_i-\sum_{j=1}^{i-1}A_{ij}x_j^{(k+1)}-\sum_{j=i+1}^{M}A_{ij}x_j^{(k)}\Big)$$

$$(i=1,2,\cdots M)$$

其中ω为**松弛因子**(Relaxation Factor)．

7.11.7 例 写出单位方域（$0\leqslant x\leqslant 1$，$0\leqslant y\leqslant 1$）上Poisson（泊松）方程

$$\begin{cases}\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=x, & 0<x<1, 0<y<1\\ u=0, & x=0,1, \quad y=0,1\end{cases}$$

的五点差分格式的线SOR的迭代公式．

解 令$h=\Delta x=\Delta y=\frac{1}{4}$,网格节点编号如图(7.11.8)所示。

7.11.8 图

	7	8	9
	4	5	6
	1	2	3

用五点差分格式逼近Poisson(泊松)方程，得方程组$Au=b$，即

$$\left(\begin{array}{ccc|ccc|ccc} 4 & -1 & 0 & -1 & & & & & \\ -1 & 4 & -1 & 0 & -1 & & & & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & & \\ & -1 & 0 & -1 & 4 & -1 & 0 & -1 & \\ & & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{array}\right)\left(\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \hline u_4 \\ u_5 \\ u_6 \\ \hline u_7 \\ u_8 \\ u_9 \end{array}\right)=\left(\begin{array}{c} \frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline \frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline \frac{1}{4} \\ \frac{1}{2} \\ 1 \end{array}\right) \quad (7.11.9)$$

按线进行分块，第一条线上的未知量u_1，u_2，u_3为第一组，记为U_1；第二条线上的未知数u_4，u_5，u_6为第二组，记为U_2，第三条线上的u_7，u_8，u_9为第三组，记为U_3。以同样的方式把右端项也分成三组。在(7.11.9)中用虚线把矩阵分成相应的块，用块矩阵表示为

$$
7.11.10 \quad \begin{pmatrix} A_{11} & A_{12} & \\ A_{21} & A_{22} & A_{23} \\ & A_{32} & A_{33} \end{pmatrix} \begin{pmatrix} U_1 \\ U_2 \\ U_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}
$$

若使用块G-S公式，具体运算按下面三组方程进行。

$$
\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_1^{(k+1)} \\ u_2^{(k+1)} \\ u_3^{(k+1)} \end{pmatrix} = -\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_4^{(k)} \\ u_5^{(k)} \\ u_6^{(k)} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \\ 1 \end{pmatrix}
$$

$$
\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_4^{(k+1)} \\ u_5^{(k+1)} \\ u_6^{(k+1)} \end{pmatrix} = -\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_1^{(k+1)} \\ u_2^{(k+1)} \\ u_3^{(k+1)} \end{pmatrix}
$$

$$
-\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_7^{(k)} \\ u_8^{(k)} \\ u_9^{(k)} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \\ 1 \end{pmatrix}
$$

$$
\begin{pmatrix} 4 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_7^{(k+1)} \\ u_8^{(k+1)} \\ u_9^{(k+1)} \end{pmatrix} = -\begin{pmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{pmatrix} \begin{pmatrix} u_4^{(k+1)} \\ u_5^{(k+1)} \\ u_6^{(k+1)} \end{pmatrix} + \begin{pmatrix} \frac{1}{4} \\ \frac{1}{2} \\ 1 \end{pmatrix}
$$

任意给定初始向量 $U^{(0)}=(u_1^{(0)},\ u_2^{(0)},\ \cdots u_9^{(0)})^T$，对于$k=0$，1，2，…求解以上三组方程，直到解达到预定精度为止。

7.11.3 块SOR迭代法的收敛性和最优松弛因子$\omega^{(*)}$的选取

块迭代法的收敛条件与逐点迭代法类似。

7.11.11 **定理** 若A为对称矩阵，A_{ii} ($i=1, 2, \cdots M$) 为正定矩阵，且 $0<\omega<2$，则逐次块（或线）SOR迭代法(7.11.6)收敛的充分必要条件为A是正定矩阵。

逐次点迭代法中的性质`A'和相容次序(7.3.3)可推广到块迭代法。把(7.11.1)中块矩阵A的子矩阵 A_{11}, A_{12}, $\cdots A_{MM}$对应新的M阶矩阵C中的元素$C_{11}, C_{12}, \cdots, C_{MM}$，则每个元素按如下方式取值

7.11.12 $$C_{ij}=\begin{cases}0, & 若A_{ij}=0\\ 1, & 若A_{ij}\neq 0\end{cases}$$

当矩阵$C=(C_{ij})_M$具有性质`A'(定义7.3.18)时，则称矩阵A具有**性质$A^{(\pi)}$**。当矩阵C有相容次序（定义7.3.24）时，则称矩阵A具有π**相容次序**（用符号π表示该种确定的分块方式）。

当对称正定矩阵A具有**性质$A^{(\pi)}$**和π**相容次序**时，逐次块SOR迭代法的最优松弛因子为

7.11.13 $$\omega_{opt}^{(\pi)}=\frac{2}{1+\sqrt{1-(\overline{\mu}^{(\pi)})^2}}$$

其中，$\overline{\mu}^{(\pi)}$为矩阵$B_J=I_n-D^{-1}A$的谱半径，D是对角元素为A_{ii}的对角阵。

相应$\omega_{opt}^{(\pi)}$的块SOR迭代矩阵$B_{SOR(\pi)}$的谱半径为

7.11.14 $$\rho(B_{SOR(\pi)})=\omega_{opt}^{(\pi)}-1$$

在下列几种情况下使用块 SOR 迭代法的实际效果比较显著：

1° 对A_{ii}作Cholesky（乔莱斯基）因式分解为LL^T后，使用块SOR迭代法，每迭代一次的工作量与点迭代相差不多，但收敛速度提高了。

2° 某些矩阵没有性质`A'，但适当分块后具有性质

$A^{(\pi)}$,例如，Laplace（拉普拉斯）方程的九点差分方程式以及重调和差分方程的相应矩阵就是这种情况。使用块SOR迭代法比点迭代法有较高的收敛速度。

3° 求解区域是狭长的带状区域。

但有时，特别是当子矩阵A_{ll}的形状复杂时，使用块SOR迭代法每次迭代的计算工作量会比点迭代增加，尽管收敛速度有提高，但总的计算时间却不见减少。

7.11.4 交替方向隐式方法

交替方向隐式方法即 ADI 法（Alternating Direction Implicit Method）。它可以看作是一种分块迭代法，其矩阵分块方式与块迭代的分块方式不同，它采用两种分块方式交替使用的办法，同时引入了加速收敛参数。ADI方法有好几种，但常用的仅是Peaceman-Rachford(俾斯曼-瑞奇福尔德）方法。

设矩阵A可以分解为$A=A_1+A_2$,且$A_1=H+\frac{1}{2}D$,$A_2=V+\frac{1}{2}D$，其中D是非负对角阵，H和V是非对角线元素为非正的对称正定阵，H和V满足可交换条件即$HV=VH$。这说明A_1和A_2是对称正定的，且$A_1A_2=A_2A_1$。从而得到与方程组

$$A\boldsymbol{u}=\boldsymbol{b}$$

等价的两个方程组

$$\text{7.11.15}\quad \begin{cases}(A_1+\alpha I_n)\boldsymbol{u}=\boldsymbol{b}-(A_2-\alpha I_n)\boldsymbol{u}\\(A_2+\alpha I_n)\boldsymbol{u}=\boldsymbol{b}-(A_1-\alpha I_n)\boldsymbol{u}\end{cases}$$

其中α为任意参数。

Peaceman-Rachford ADI迭代公式是

$$\text{7.11.16}\quad (A_1+\alpha I_n)\boldsymbol{u}^{(k+\frac{1}{2})}=\boldsymbol{b}-(A_2-\alpha I_n)\boldsymbol{u}^{(k)}$$

7.11.17 $(A_2+\alpha I_n)\boldsymbol{u}^{(k+1)}=\boldsymbol{b}-(A_1-\alpha I_n)\boldsymbol{u}^{(k+\frac{1}{2})}$

其中 α 为**加速参数**(Acceleration Parameter)。通常$A_1+\alpha I_n$是三对角矩阵，$A_2+\alpha I_n$经适当变换后也是三对角矩阵。

ADI迭代法的计算过程是从任意$\boldsymbol{u}^{(0)}$出发（在例7.11.25中可以看到），先沿水平方向逐线隐式求解三对角方程组(7.11.16)，得$\boldsymbol{u}^{(k+\frac{1}{2})}$，然后沿垂直方向逐线隐式求解由(7.11.17)经变换以后得到的三对角方程组，得$\boldsymbol{u}^{(k+1)}$.两步合起来才算完成一次迭代，故称**交替方向隐式迭代法**。

由（7.11.16）和（7.11.17）两式合成一个式子为

7.11.18 $\boldsymbol{u}^{(k+1)}=B_{ADI}\boldsymbol{u}^{(k)}+\boldsymbol{f}_{ADI}$

其中

7.11.19 $B_{ADI}=(A_2+\alpha I_n)^{-1}(A_1-\alpha I_n)(A_1+\alpha I_n)^{-1}(A_2-\alpha I_n)$

称为**ADI迭代矩阵**。

设μ是矩阵A_1的特征值$0<a_1\leqslant\mu\leqslant d_1$，$\lambda$是矩阵$A_2$的特征值 $0<a_2\leqslant\lambda\leqslant d_2$，$\chi$是矩阵$B_{ADI}$的特征值，则有

7.11.20 $\chi=\dfrac{(\mu-\alpha)(\lambda-\alpha)}{(\mu+\alpha)(\lambda+\alpha)}$

选取加速参数α使χ达到最小，得

7.11.21 $\alpha=\sqrt{ad}$

其中 $a=\min(a_1,\ a_2)$，$d=\max(d_1,d_2)$.

用可变加速参数 α_k 更加有效，可变加速参数的 Peaceman-Rachford ADI公式是

7.11.22 $(A_1+\alpha_{k+1}I_n)\boldsymbol{u}^{(k+\frac{1}{2})}=\boldsymbol{b}-(A_2-\alpha_{k+1}I_n)\boldsymbol{u}^{(k)}$

7.11.23 $(A_2+\alpha_{k+1}I_n)\boldsymbol{u}^{(k+1)}=\boldsymbol{b}-(A_1-\alpha_{k+1}I_n)\boldsymbol{u}^{(k+\frac{1}{2})}$

其中

7.11.24 $\alpha_k=d(a/d)^{(2k-1)/2k}\quad(k=1,2,\cdots)$

当$k=1$时，上式就是（7.11.21）。对于α_k，采用循环使用的办法α_1，$\alpha_2\cdots$，α_k；α_1，α_2，$\cdots\alpha_k\cdots$。

7.11.25 **例** 写出单位正方形域（$0\leqslant x\leqslant 1$，$0\leqslant y\leqslant 1$）上

的Poisson（泊松）方程

$$\begin{cases} \dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}=x, & 0<x<1,0<y<1 \\ u=0, & x=0,1, \quad y=0,1 \end{cases}$$

五点差分格式的ADI迭代公式。

解 令 $h=\Delta x=\Delta y=\dfrac{1}{4}$，网格节点编号如图(7.11.26)。

7.11.26 图

	7	8	9
	4	5	6
	1	2	3

用五点差分格式逼近Poisson方程，得方程组

7.11.27 $Au=b$，即

$$\left(\begin{array}{ccc|ccc|ccc} 4 & -1 & 0 & -1 & & & & & \\ -1 & 4 & -1 & 0 & -1 & & & & \\ 0 & -1 & 4 & 0 & 0 & -1 & & & \\ \hline -1 & 0 & 0 & 4 & -1 & 0 & -1 & & \\ & -1 & 0 & -1 & 4 & -1 & 0 & -1 & \\ & & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ \hline & & & -1 & 0 & 0 & 4 & -1 & 0 \\ & & & & -1 & 0 & -1 & 4 & -1 \\ & & & & & -1 & 0 & -1 & 4 \end{array}\right)\left(\begin{array}{c} u_1 \\ u_2 \\ u_3 \\ \hline u_4 \\ u_5 \\ u_6 \\ \hline u_7 \\ u_8 \\ u_9 \end{array}\right)=\left(\begin{array}{c} \frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline \frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline \frac{1}{4} \\ \frac{1}{2} \\ 1 \end{array}\right) \tag{7.11.28}$$

将矩阵A进行分块，使$A=A_1+A_2$，

$$
A_1=\begin{pmatrix}
2 & -1 & & & & & & & \\
-1 & 2 & -1 & & & & & & \\
 & -1 & 2 & & & & & & \\
 & & & 2 & -1 & & & & \\
 & & & -1 & 2 & -1 & & & \\
 & & & & -1 & 2 & & & \\
 & & & & & & 2 & -1 & \\
 & & & & & & -1 & 2 & -1 \\
 & & & & & & & -1 & 2
\end{pmatrix} \tag{7.11.29}
$$

$$
A_2=\begin{pmatrix}
2 & 0 & 0 & -1 & & & & & \\
0 & 2 & 0 & 0 & -1 & & & & \\
0 & 0 & 2 & 0 & 0 & -1 & & & \\
-1 & 0 & 0 & 2 & 0 & 0 & -1 & & \\
 & -1 & 0 & 0 & 2 & 0 & 0 & -1 & \\
 & & -1 & 0 & 0 & 2 & 0 & 0 & -1 \\
 & & & -1 & 0 & 0 & 2 & 0 & 0 \\
 & & & & -1 & 0 & 0 & 2 & 0 \\
 & & & & & -1 & 0 & 0 & 2
\end{pmatrix}
$$

其中$A_1=H+\frac{1}{2}D$，$A_2=V+\frac{1}{2}D$。 D是矩阵A的对角线元素组成的对角阵，H和V都是非对角线元素为非正的对称正定阵，且满足可交换条件即$HV=VH$，因而A_1和A_2也是对称正定的，且可交换$A_1A_2=A_2A_1$。根据ADI迭代公式(7.11.16)，有求解（7.11.28）的x方向的迭代公式

$$
\textbf{7.11.30}\quad
\left(\begin{array}{ccc|ccc|ccc}
2+\alpha & -1 & & & & & & & \\
-1 & 2+\alpha & -1 & & & & & & \\
 & -1 & 2+\alpha & & & & & & \\
\hline
 & & & 2+\alpha & -1 & & & & \\
 & & & -1 & 2+\alpha & -1 & & & \\
 & & & & -1 & 2+\alpha & & & \\
\hline
 & & & & & & 2+\alpha & -1 & \\
 & & & & & & -1 & 2+\alpha & -1 \\
 & & & & & & & -1 & 2+\alpha
\end{array}\right)
\left(\begin{array}{c}
u_1^{(k+\frac{1}{2})} \\ u_2^{(k+\frac{1}{2})} \\ u_3^{(k+\frac{1}{2})} \\ \hline
u_4^{(k+\frac{1}{2})} \\ u_5^{(k+\frac{1}{2})} \\ u_6^{(k+\frac{1}{2})} \\ \hline
u_7^{(k+\frac{1}{2})} \\ u_8^{(k+\frac{1}{2})} \\ u_9^{(k+\frac{1}{2})}
\end{array}\right)
$$

$$
=\left(\begin{array}{c}
\frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline
\frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \hline
\frac{1}{4} \\ \frac{1}{2} \\ 1
\end{array}\right)
-
\left(\begin{array}{ccc|ccc|ccc}
2-\alpha & 0 & 0 & -1 & & & & & \\
0 & 2-\alpha & 0 & 0 & -1 & & & & \\
0 & 0 & 2-\alpha & 0 & 0 & -1 & & & \\
\hline
-1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 & & \\
 & -1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 & \\
 & & -1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 \\
\hline
 & & & -1 & 0 & 0 & 2-\alpha & 0 & 0 \\
 & & & & -1 & 0 & 0 & 2-\alpha & 0 \\
 & & & & & -1 & 0 & 0 & 2-\alpha
\end{array}\right)
\left(\begin{array}{c}
u_1^{(k)} \\ u_2^{(k)} \\ u_3^{(k)} \\ \hline
u_4^{(k)} \\ u_5^{(k)} \\ u_6^{(k)} \\ \hline
u_7^{(k)} \\ u_8^{(k)} \\ u_9^{(k)}
\end{array}\right)
$$

水平方向有三条线：节点1,2,3在l_1上；节点4,5,6在l_2上；节点7,8,9在l_3上．按这三条线可以把上式矩阵和向量用虚线进行分块逐线解出$u^{(k+\frac{1}{2})}$．

再根据（7.11.17），有y方向的迭代公式

$$
(7.11.31)\quad
\begin{pmatrix}
2+\alpha & 0 & 0 & -1 & & & & & \\
0 & 2+\alpha & 0 & 0 & -1 & & & & \\
0 & 0 & 2+\alpha & 0 & 0 & -1 & & & \\
-1 & 0 & 0 & 2+\alpha & 0 & 0 & -1 & & \\
 & -1 & 0 & 0 & 2+\alpha & 0 & 0 & -1 & \\
 & & -1 & 0 & 0 & 2+\alpha & 0 & 0 & -1 \\
 & & & -1 & 0 & 0 & 2+\alpha & 0 & 0 \\
 & & & & -1 & 0 & 0 & 2+\alpha & 0 \\
 & & & & & -1 & 0 & 0 & 2+\alpha
\end{pmatrix}
\begin{pmatrix}
u_1^{(k+1)} \\ u_2^{(k+1)} \\ u_3^{(k+1)} \\ u_4^{(k+1)} \\ u_5^{(k+1)} \\ u_6^{(k+1)} \\ u_7^{(k+1)} \\ u_8^{(k+1)} \\ u_9^{(k+1)}
\end{pmatrix}
$$

$$
=\begin{pmatrix}
\frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \frac{1}{4} \\ \frac{1}{2} \\ 1 \\ \frac{1}{4} \\ \frac{1}{2} \\ 1
\end{pmatrix}
-\begin{pmatrix}
2-\alpha & -1 & & & & & & & \\
-1 & 2-\alpha & -1 & & & & & & \\
 & -1 & 2-\alpha & & & & & & \\
 & & & 2-\alpha & -1 & & & & \\
 & & & -1 & 2-\alpha & -1 & & & \\
 & & & & -1 & 2-\alpha & & & \\
 & & & & & & 2-\alpha & -1 & \\
 & & & & & & -1 & 2-\alpha & -1 \\
 & & & & & & & -1 & 2-\alpha
\end{pmatrix}
\begin{pmatrix}
u_1^{(k+\frac{1}{2})} \\ u_2^{(k+\frac{1}{2})} \\ u_3^{(k+\frac{1}{2})} \\ u_4^{(k+\frac{1}{2})} \\ u_5^{(k+\frac{1}{2})} \\ u_6^{(k+\frac{1}{2})} \\ u_7^{(k+\frac{1}{2})} \\ u_8^{(k+\frac{1}{2})} \\ u_9^{(k+\frac{1}{2})}
\end{pmatrix}
$$

事实上，并非按上式求解．按y方向的线S_1上有节点1,4,7，在线S_2上有节点2,5,8；在线S_3上有节点3，6，9，按这种节点次序将方程组进行行列交换后，得（7.11.32）的迭代公式

$$
\begin{pmatrix}
2+\alpha & -1 & & & & & & & \\
-1 & 2+\alpha & -1 & & & & & & \\
 & -1 & 2+\alpha & & & & & & \\
 & & & 2+\alpha & -1 & & & & \\
 & & & -1 & 2+\alpha & -1 & & & \\
 & & & & -1 & 2+\alpha & & & \\
 & & & & & & 2+\alpha & -1 & \\
 & & & & & & -1 & 2+\alpha & -1 \\
 & & & & & & & -1 & 2+\alpha
\end{pmatrix}
\begin{pmatrix}
u_1^{(k+1)} \\ u_4^{(k+1)} \\ u_7^{(k+1)} \\ u_2^{(k+1)} \\ u_5^{(k+1)} \\ u_8^{(k+1)} \\ u_3^{(k+1)} \\ u_6^{(k+1)} \\ u_9^{(k+1)}
\end{pmatrix} \tag{7.11.32}
$$

$$
=\begin{pmatrix}
1/4 \\ 1/4 \\ 1/4 \\ 1/2 \\ 1/2 \\ 1/2 \\ 1 \\ 1 \\ 1
\end{pmatrix}
-\begin{pmatrix}
2-\alpha & 0 & 0 & -1 & & & & & \\
0 & 2-\alpha & 0 & 0 & -1 & & & & \\
0 & 0 & 2-\alpha & 0 & 0 & -1 & & & \\
-1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 & & \\
 & -1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 & \\
 & & -1 & 0 & 0 & 2-\alpha & 0 & 0 & -1 \\
 & & & -1 & 0 & 0 & 2-\alpha & 0 & 0 \\
 & & & & -1 & 0 & 0 & 2-\alpha & 0 \\
 & & & & & -1 & 0 & 0 & 2-\alpha
\end{pmatrix}
\begin{pmatrix}
u_1^{(k+\frac{1}{2})} \\ u_4^{(k+\frac{1}{2})} \\ u_7^{(k+\frac{1}{2})} \\ u_2^{(k+\frac{1}{2})} \\ u_5^{(k+\frac{1}{2})} \\ u_8^{(k+\frac{1}{2})} \\ u_3^{(k+\frac{1}{2})} \\ u_6^{(k+\frac{1}{2})} \\ u_9^{(k+\frac{1}{2})}
\end{pmatrix}
$$

此方程组的解法和（7.11.30）一样，只是沿 y方向的线逐线解出 $u^{(k+1)}$。反复按（7.11.30）和（7.11.32）进行计算，直到近似解达到预先给定的精度为止。

由于方程组（7.11.30）和（7.11.32）中左边都是矩阵 $A_1+\alpha I_n$，右边都是$A_2-\alpha I_n$，所不同的是未知向量和右端项各分量的排列次序不同，因此，用 ADI 迭代法解方域上的Poisson方程是比较简单的，而且收敛速度得到提高。

7.12

7.12 不完全LU分解法和强隐式方法

7.12.1 不完全LU分解法

线性代数方程组

7.12.1 $Au=F$

其系数矩阵A是大型稀疏矩阵。为求解方程组，把A分解成

7.12.2 $A=LU-R$

使L和U分别为下三角阵和上三角阵，且和A的下三角部分和上三角部分有相同的稀疏结构；A和L，U可以使用相同的存储区域。同时要求

7.12.3 $(LU)^{-1}\geqslant 0$

7.12.4 $R\geqslant 0$

这样的分解称为**不完全LU分解**（Incomplete LU Factorization）。

不完全LU分解的迭代公式是

7.12.5 $u^{(k+1)}=(LU)^{-1}Ru^{(k)}+(LU)^{-1}F$

只要$A^{-1}\geqslant 0$，且条件（7.12.3)和(7.12.4)成立，则(7.12.5)迭代公式收敛。

每一步迭代都要求解形如

7.12.6 $LUu^{(k+1)}=Ru^{(k)}+F\equiv G$

的方程组。由于L和U分别是下三角阵和上三角阵，所以很容易直接求解。

构造L和U的整个过程由$N-1$步组成，N是方程组的阶数。首先，用一个预先给定的集合

7.12.7 $P=\{(i,j)\mid i\neq j \quad 1\leqslant i,j\leqslant N\}$

表示L和U中的零元素位置。再由下列关系式定义一系列的矩阵A_k，$\tilde{A}_k$，L_k和R_k：

7.12.8 $A_0=A$

7.12.9 $\begin{cases}\widetilde{A}_k = A_{k-1} + R_k & (k=1,2,\cdots,N-1)\\ A_k = L_k\widetilde{A}_k & (k=1,2,\cdots,N-1)\end{cases}$

7.12.10 $A_{N-1} = U$

R_k由下列各式确定

7.12.11 $\begin{cases} r^k_{k,j} = -a^{k-1}_{k,j}, & (k,j)\in P\\ r^k_{i,k} = -a^{k-1}_{i,k}, & (i,k)\in P\\ r^k_{i,j} = 0, & 其它的(i,j)\end{cases}$

7.12.12 $L_k = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & -\dfrac{\widetilde{a}^k_{k+1,k}}{\widetilde{a}^k_{kk}} & 1 & & \\ & & \vdots & & \ddots & \\ & & -\dfrac{\widetilde{a}^k_{N,k}}{\widetilde{a}^k_{kk}} & & & 1 \end{pmatrix}$

则

$$L_k^{-1} = \begin{pmatrix} 1 & & & & & \\ & \ddots & & & & \\ & & 1 & & & \\ & & \dfrac{\widetilde{a}^k_{k+1,k}}{\widetilde{a}^k_{kk}} & 1 & & \\ & & \vdots & & \ddots & \\ & & \dfrac{\widetilde{a}^k_{N,k}}{\widetilde{a}^k_{kk}} & & & 1 \end{pmatrix}$$

由以上各矩阵的定义有：

1° $\widetilde{A}_k$是把 A_{k-1} 中那些零元素集合P相应位置上出现的非零元素置零后得到的矩阵。

2° A_k是由$\widetilde{A}_k$的第 k 列经过Gauss(高斯)消去后 得到的矩阵。A_{N-1}中凡下标属于 P 的元素都已变为零。

3° $R_k \geqslant 0(k=1,2, \cdots, N-1)$, R_k仅在第 k 行和 第 k 列上有非零元素，其余元素都是零。

4° L_k^{-1}除对角线元素是1外，仅在第k列中有非零元素。凡下标属于 P 的元素都为零，$L_k \geqslant 0$，所以

$L_1^{-1} R_2 = R_2$

同理，有

7.12.13 $$\begin{cases} L_1^{-1} L_2^{-1} R_3 = R_3 \\ \cdots \\ L_1^{-1} L_2^{-1} \cdots L_{N-2}^{-1} R_{N-1} = R_{N-1} \end{cases}$$

由 (7.12.10) 和 (7.12.9)，有

7.12.14 $$\begin{aligned} U &= (L_{N-1} L_{N-2} \cdots L_1)(A_0 + R_1 + R_2 + \cdots + R_{N-1}) \\ &= (L_{N-1} L_{N-2} \cdots L_1)(A + R) \end{aligned}$$

由 (7.12.2) 式，得

7.12.15 $$\begin{aligned} L &= (L_{N-1} L_{N-2} \cdots L_1)^{-1} = L_1^{-1} L_2^{-1} \cdots L_{N-1}^{-1} \\ &= I + \widetilde{L}_1 + \widetilde{L}_2 + \cdots + \widetilde{L}_{N-1} \end{aligned}$$

其中$\widetilde{L}_i$是L_i^{-1}的严格下三角部分。又有

7.12.16 $$R = R_1 + R_2 + \cdots + R_{N-1} \geqslant 0$$

由于

$(LU)^{-1} = U^{-1} L_{N-1} L_{N-2} \cdots L_1$

右端每个因子都是非负阵，因此

$(LU)^{-1} \geqslant 0$

这样的分解确是不完全LU分解。

7.12.17 **推论** 如果A是对称的，P是对称的零元素位置集合，即$(i,j) \in P$，必有$(j, i) \in P$，则

$A = LL^T - R$

7.12.18 例 对矩阵

$$A=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ -1 & 4 & -1 & 0 & -1 \\ 0 & -1 & 4 & -1 & 0 \\ -1 & 0 & -1 & 4 & -1 \\ 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

作不完全LU分解。

解 因为 A 是对称的，可以将 A 分解成

$A=LL^T-R$

为使 L 和 A 的下三角部分保持相同的非零结构，故预先给定零元素集合

$P=\{(i,\ j)\mid |i-j|\neq 0,\ 1,\ 3\}$

不完全LU分解过程如下：

$A_0=A$, $R_1=0$, $A_1=A$

$$L_1=\begin{pmatrix} 1 & & & & \\ \frac{1}{4} & 1 & & & \\ 0 & & 1 & & \\ \frac{1}{4} & & & 1 & \\ 0 & & & & 1 \end{pmatrix},\quad A_1=L_1A_0=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ 0 & \frac{15}{4} & -1 & -\frac{1}{4} & -1 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & -\frac{1}{4} & -1 & \frac{15}{4} & -1 \\ 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

$$R_2=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & \frac{1}{4} & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & \frac{1}{4} & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{pmatrix},\quad \widetilde{A_2}=A_1+R_2=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ 0 & \frac{15}{4} & -1 & 0 & -1 \\ 0 & -1 & 4 & -1 & 0 \\ 0 & 0 & -1 & \frac{15}{4} & -1 \\ 0 & -1 & 0 & -1 & 4 \end{pmatrix}$$

$$L_2=\begin{pmatrix} 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 \\ 0 & \frac{4}{15} & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 \\ 0 & \frac{4}{15} & 0 & 0 & 1 \end{pmatrix}, \quad A_2=L_2\widetilde{A}_2=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ 0 & \frac{15}{4} & -1 & 0 & -1 \\ 0 & 0 & \frac{56}{15} & -1 & -\frac{4}{15} \\ 0 & 0 & -1 & \frac{15}{4} & -1 \\ 0 & 0 & -\frac{4}{15} & -1 & \frac{56}{15} \end{pmatrix}$$

$$R_3=\begin{pmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{4}{15} \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & \frac{4}{15} & 0 & 0 \end{pmatrix}, \quad \widetilde{A}_3=A_2+R_3=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ 0 & \frac{15}{4} & -1 & 0 & -1 \\ 0 & 0 & \frac{56}{15} & -1 & 0 \\ 0 & 0 & -1 & \frac{15}{4} & -1 \\ 0 & 0 & 0 & -1 & \frac{56}{15} \end{pmatrix}$$

$$L_3=\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & \frac{15}{56} & 1 & \\ & & & & 1 \end{pmatrix}, \quad A_3=L_3\widetilde{A}_3=\begin{pmatrix} 4 & -1 & 0 & -1 & 0 \\ 0 & \frac{15}{4} & -1 & 0 & -1 \\ 0 & 0 & \frac{56}{15} & -1 & 0 \\ 0 & 0 & 0 & \frac{13\times 15}{56} & -1 \\ 0 & 0 & 0 & -1 & \frac{56}{15} \end{pmatrix}$$

$$R_4=0 \quad , \qquad \widetilde{A}_4=A_3$$

$$L_4=\begin{pmatrix} 1 & & & & \\ & 1 & & & \\ & & 1 & & \\ & & & 1 & \\ & & & \frac{56}{13\times 15} & 1 \end{pmatrix}$$

$$
A_4=L_4\widetilde{A}_4=\begin{pmatrix}
4 & -1 & 0 & -1 & 0 \\
 & \frac{15}{4} & -1 & 0 & -1 \\
 & & \frac{56}{15} & -1 & 0 \\
 & & & \frac{13\times 15}{56} & -1 \\
 & & & & \frac{56}{15}-\frac{56}{13\times 15}
\end{pmatrix}=U
$$

$$
L=I+\widetilde{L}_1+\widetilde{L}_2+\widetilde{L}_3+\widetilde{L}_4=\begin{pmatrix}
1 & & & & \\
\frac{1}{4} & 1 & & & \\
0 & \frac{4}{15} & 1 & & \\
\frac{1}{4} & 0 & \frac{15}{56} & 1 & \\
0 & \frac{4}{15} & 0 & \frac{56}{13\times 15} & 1
\end{pmatrix}
$$

$$
R=R_1+R_2+R_3+R_4=\begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{1}{4} & 0 \\
0 & 0 & 0 & 0 & \frac{4}{15} \\
0 & \frac{1}{4} & 0 & 0 & 0 \\
0 & 0 & \frac{4}{15} & 0 & 0
\end{pmatrix}
$$

7.12.2 强隐式方法

强隐式方法（Strongly Implicit Method）比SOR迭代法（7.7）和ADI迭代法（7.11.4）的收敛速度快。

用迭代求解线性方程组

7.12.19 $Ax=s$

时，在此式两边加上辅助项$\widetilde{A}x$，得到迭代公式

7.12.20 $(A+\widetilde{A})x^{(k+1)}=\widetilde{A}x^{(k)}+s$

选取的辅助矩阵$\widetilde{A}$应当使$A+\widetilde{A}$容易求逆，故常用的一些迭代公式都是（7.12.20）的特例．例如，如果将A进行分解使$A=D-L-U$，D是对角阵，L为严格下三角阵．U为严格上三角阵，当$\widetilde{A}=L+U$时，（7.12.20)就是Jacobi迭代公式；当$\widetilde{A}=U$时，（7.12.20）就是Gauss-Seidel迭代公式．一般情况下，$A+\widetilde{A}$愈逼近于A，收敛愈快．

强隐式方法是选取辅助矩阵$\widetilde{A}$，使$A+\widetilde{A}$能够进行因式分解成

$$A+\widetilde{A}=LU$$

其中L和U分别是下三角阵和上三角阵，且与A的下三角部分和上三角部分有相同的稀疏结构，或者说非零元素分布形状相同．LU形式的矩阵很容易求逆．如果$\widetilde{A}$是对A的一个小的修正，则称（7.12.20）表示的迭代格式**为强隐式格式**．

7.12.21 $(A+\widetilde{A})x^{(k+1)}=(A+\widetilde{A})x^{(k)}-\omega(Ax^{(k)}-s)$

的迭代格式，其中$\widetilde{A}$是辅助矩阵，ω是每次迭代都可以改变的迭代参数，而迭代矩阵

$$B_{SI}=I_n-\omega(A+\widetilde{A})^{-1}A$$

7.12.22 **定理** 如果A和$A+\widetilde{A}$**是正定的**（定义7.3.10），则当且仅当

$$0<\omega<\frac{2}{\lambda_{max}[(A+\widetilde{A})^{-1}A]}$$

时，强隐式迭代（7.12.21）是收敛的．

7.12.23 **定理** 如果A和$A+\widetilde{A}$是正定的，则迭代参数ω的最优值是

$$\omega_{opt}=\frac{2}{\lambda_{max}[(A+\widetilde{A})^{-1}A]+\lambda_{min}[(A+\widetilde{A})^{-1}A]}$$

其中$\lambda_{max}[C]$和$\lambda_{min}[C]$是矩阵 C 的最大特征值和 最 小 特征值。

实际上，$\lambda_{max}[(A+\widetilde{A})^{-1}A]$和$\lambda_{min}[(A+\widetilde{A})^{-1}A]$是 不 知道的。Diamond提出了一种在迭代过程中寻求ω_{opt}的 办法；迭代开始给一个粗的估计 值，如$\lambda_{max}[(A+\widetilde{A})^{-1}A]=a=2.5$，$\lambda_{min}[(A+\widetilde{A})^{-1}A]=b=0.5$，在迭代过程中观察余量再对 a 和 b作修改。

对于二阶椭圆型偏微分方程

$$a\frac{\partial^2 u}{\partial x^2}+b\frac{\partial^2 u}{\partial y^2}+c\frac{\partial u}{\partial x}+d\frac{\partial u}{\partial y}+gu=f$$

相应的五点差分格式为

7.12.24 $a^L_{ij}u_{i-1,j}+a^B_{ij}u_{i,j-1}+a^R_{ij}u_{i+1,j}$
$+a^T_{ij}u_{i,j+1}-a^C_{ij}u_{ij}=s_{ij}$

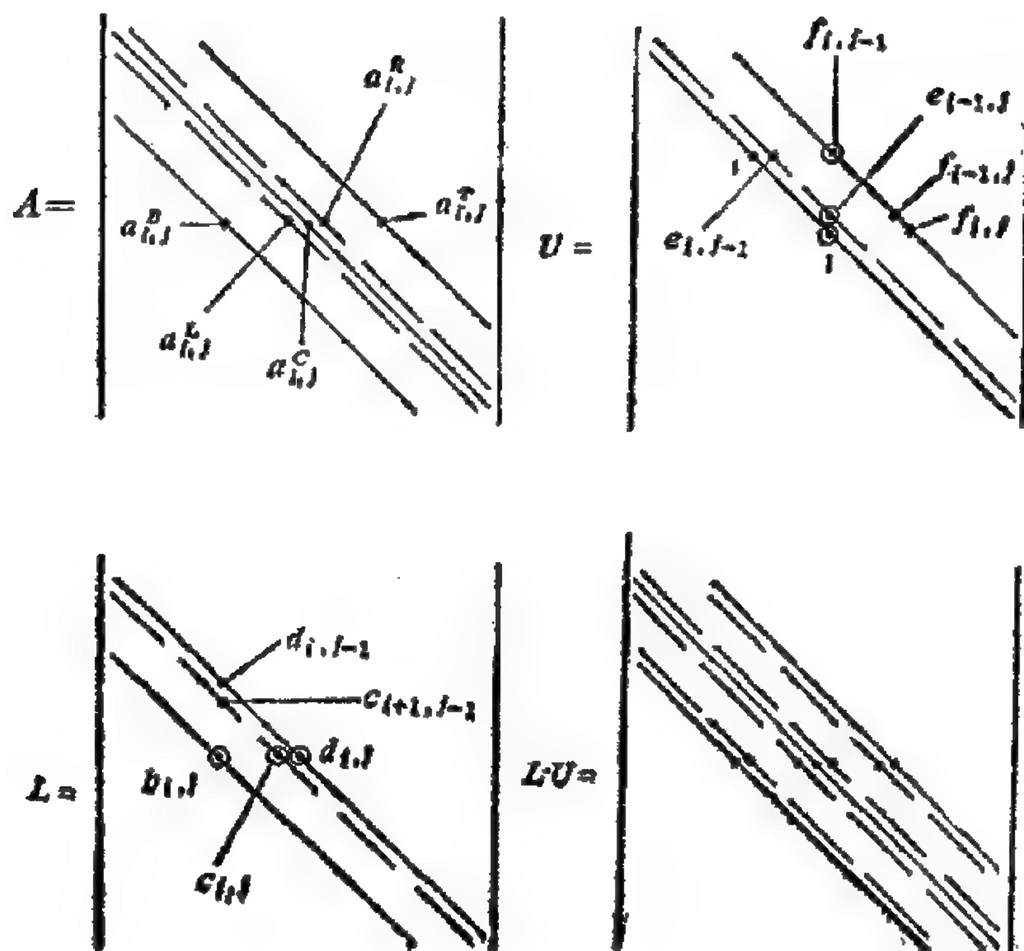
用矩阵形式表示，有

7.12.25 $Au=s$

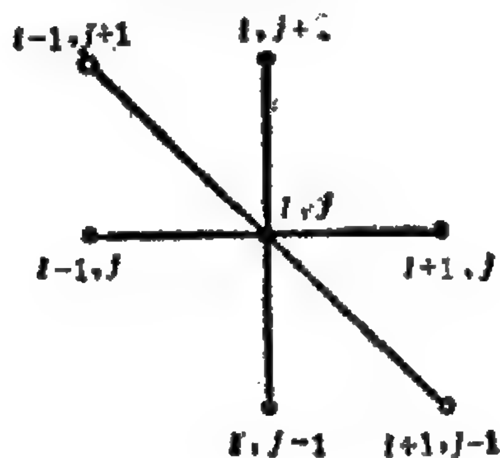
其中系数矩阵 A 是**五对角**的（Pentadiagonal），L 和U 的结构分别与 A 的下三角部分和上三角部分的稀疏结构形状相同，L 和U 的乘积变成**七对角阵**(Heptadiagonal Matrix)，如图（7.12.26）。LU和 A 相比每行增加了二个元 素，在 图（7.12.27）中用○表示。

为了使$A+\widetilde{A}=LU$，则$\widetilde{A}$ 必定是七对 角 阵，用$\widetilde{a}^L_{ij}$，$\widetilde{a}^{BR}_{ij}$，$\widetilde{a}^T_{ij}$，$\widetilde{a}^C_{ij}$，$\widetilde{a}^B_{ij}$，$\widetilde{a}^{TL}_{ij}$和$\widetilde{a}^R_{ij}$分别表示 $\widetilde{A}$一行中从左到右的七个元素。LU和$A+\widetilde{A}$的元素之间一定满足如下关系：

7.12.26 图 强隐式方法矩阵结构



7.12.27 图 强隐式方法的格式



7.12.28 $b_{ij}=a^B_{ij}+\widetilde{a}^B_{ij}$

7.12.29 $b_{ij}e_{i,j-1}=\widetilde{a}^{BR}_{ij}$

7.12.30 $c_{ij}=a^L_{ij}+\widetilde{a}^L_{ij}$

7.12.31 $d_{ij}+b_{ij}f_{i,j-1}+c_{ij}e_{i-1,j}=a^C_{ij}+\widetilde{a}^C_{ij}$

7.12.32 $d_{ij}e_{ij}=a^R_{ij}+\widetilde{a}^R_{ij}$

7.12.33 $c_{ij}f_{i-1,j}=\widetilde{a}^{TL}_{ij}$

7.12.34 $d_{ij}f_{ij}=a^T_{ij}+\widetilde{a}^T_{ij}$

任意选定一个$\widetilde{A}$并不能唯一地确定L和U，因为在上面七个方程中只有五个未知量b_{ij},c_{ij},d_{ij}，e_{ij}，f_{ij}，显然有二个方程是线性相关的。先删去（7.12.29）和（7.12.33），由其它五个方程求出b_{ij}，c_{ij}后再用（7.12.29）和（7.12.33）计算$\widetilde{a}^{BR}$和$\widetilde{a}^{TL}$.因此$\widetilde{A}$必须具有以上性质，才能唯一确定L和U的元素。

选取$\widetilde{A}$的原则是使$A+\widetilde{A}$尽可能逼近A,选取办法有多种。如，用Taylor（泰勒）级数近似展开$u_{i+1,j-1}$和$u_{i-1,j+1}$，有二个近似式：

7.12.35 $u_{i+1,j-1}=-u_{ij}+u_{i,j-1}+u_{i+1,j}$

7.12.36 $u_{i-1,j+1}=-u_{ij}+u_{i,j+1}+u_{i-1,j}$

由于Taylor展开有截断误差，上两式不是精确的，故引入参数α起调节误差的作用，有

7.12.37 $u_{i+1,j-1}=\alpha(-u_{ij}+u_{i,j-1}+u_{i+1,j})$

7.12.38 $u_{i-1,j+1}=\alpha(-u_{ij}+u_{i,j+1}+u_{i-1,j})$

用$\widetilde{a}^{BR}_{ij}$乘（7.12.37），用a^{TL}_{ij}乘（7.12.38），然后分别与五点差分格式（7.12.24）等号两边相加，所得七点方程的左边是

7.12.39 $(a^B_{ij}-\alpha\widetilde{a}^{BR}_{ij})u_{i,j-1}+(a^L_{ij}-\alpha a^{TL}_{ij})u_{i-1,j}$

$$+[a_{ij}^{C}+\alpha(\widetilde{a}_{ij}^{BR}+\widetilde{a}_{ij}^{TL})]u_{ij}+(a_{ij}^{R}-\alpha\widetilde{a}_{ij}^{BR})u_{i+1,j}$$

$$+(a_{ij}^{T}-\alpha\widetilde{a}_{ij}^{TL})u_{i,j+1}+\widetilde{a}_{ij}^{BR}u_{i+1,j-1}+\widetilde{a}_{ij}^{TL}u_{i-1,j+1}$$

把它和（7.12.28），（7.12.30），（7.12.31），（7.12.32），（7.12.34）对照，有

7.12.40 $b_{ij}=a_{ij}^{B}-\alpha\widetilde{a}_{ij}^{BR}$

7.12.41 $c_{ij}=a_{ij}^{L}-\alpha\widetilde{a}_{ij}^{TL}$

7.12.42 $d_{ij}+b_{ij}f_{i,j-1}+c_{ij}e_{i-1,j}=a_{ij}^{C}+\alpha(\widetilde{a}_{ij}^{BR}+\widetilde{a}_{ij}^{TL})$

7.12.43 $d_{ij}e_{ij}=a_{ij}^{R}-\alpha\widetilde{a}_{ij}^{BR}$

7.12.44 $d_{ij}f_{ij}=a_{ij}^{T}-\alpha\widetilde{a}_{ij}^{TL}$

用（7.12.29）和（7.12.33）代入（7.12.40）－（7.12.44），消去$\widetilde{a}_{ij}^{BR}$和$\widetilde{a}_{ij}^{TL}$得

7.12.45 $b_{ij}+\alpha b_{ij}e_{i,j-1}=a_{ij}^{B}$

7.12.46 $c_{ij}+\alpha c_{ij}f_{i-1,j}=a_{ij}^{L}$

7.12.47 $d_{ij}+b_{ij}f_{i,j-1}+c_{ij}e_{i-1,j}-\alpha(b_{ij}e_{i,j-1}+c_{ij}f_{i-1,j})=a_{ij}^{C}$

7.12.48 $d_{ij}e_{ij}+\alpha b_{ij}e_{i,j-1}=a_{ij}^{R}$

7.12.49 $d_{ij}f_{ij}+\alpha c_{ij}f_{i-1,j}=a_{ij}^{T}$

其中$0<\alpha\leqslant 1$. 上面五个方程右边是A的元素为已知值，且$e_{1,0}=e_{0,1}=f_{1,0}=f_{0,1}=0$，从$i=j=1$开始，先计算$e_{i-1,j}$，$e_{i,j-1}$，$f_{i-1,j}$，$f_{i,j-1}$，然后循环地依次计算$b$，$c$，$d$，$e$和$f$，按$i$和$j$增加的次序进行，这样可以求得全部$b_{ij}$，$c_{ij}$，$d_{ij}$，$e_{ij}$和$f_{ij}$，从而得到矩阵$L$和$U$。

方程（7.12.21）可以写成

7.12.50 $(A+\widetilde{A})\delta^{(k+1)}=r^{(k)}$

其中

7.12.51 $\delta^{(k+1)}=u^{(k+1)}-u^{(k)}$

7.12.52 $r^{(k)}=-\omega(Au^{(k)}-s)$

强隐式方法的计算步骤：

1° 按（7.12.45）-（7.12.49）对$A+\widetilde{A}=LU$进行因式分解，从而算得矩阵L和U；

2° 任意选取初始值$\boldsymbol{u}^{(0)}$；

3° 计算$\boldsymbol{r}^{(k)}=-\omega(A\boldsymbol{u}^{(k)}-\boldsymbol{s})$；

4° 解方程组

$$L\boldsymbol{v}^{(k)}=\boldsymbol{r}^{(k)}$$

因为L是下三角阵，容易计算$\boldsymbol{v}^{(k)}$，即

$$v_{ij}^{(k)}=\frac{1}{d_{ij}}(r_{i,j}^{(k)}-b_{i,j}v_{i,j-1}^{(k)}-c_{i,j}v_{i-1,j}^{(k)})$$

$$(i,j=1,2,\cdots n)$$

5° 解方程组

$$U\boldsymbol{\delta}^{(k+1)}=\boldsymbol{v}^{(k)}$$

因为U是上三角阵，容易计算$\boldsymbol{\delta}^{(k+1)}$，即

$$\delta_{ij}^{(k+1)}=v_{ij}^{(k)}-e_{ij}\delta_{i+1,j}^{(k+1)}-f_{ij}\delta_{i,j+1}^{(k+1)}$$

$$(i,j=n,n-1,\cdots 1)$$

6° 计算$\boldsymbol{u}^{(k+1)}=\boldsymbol{u}^{(k)}+\boldsymbol{\delta}^{(k+1)}$；

7° 判断：若$\|\boldsymbol{\delta}^{(k+1)}\|<\varepsilon$（预先给定的小数）成立，则$\boldsymbol{u}^{(k+1)}$为（7.12.25）的解，否则重复3°—7°。

7.13

7.13 最速下降法和共轭斜量法

7.13.1 等价极小值问题

若方程组

7.13.1 $$A\boldsymbol{u}=\boldsymbol{b} \quad 或 \quad \sum_{j=1}^{n}a_{ij}u_j=b_i \quad (i=1,2,\cdots,n)$$

的系数矩阵A是对称正定的，则令

7.13.2 $$F(\boldsymbol{x})=\frac{1}{2}(A\boldsymbol{x},\boldsymbol{x})-(\boldsymbol{b},\boldsymbol{x})$$

$$=\frac{1}{2}\sum_{i=1}^{n}\sum_{j=1}^{n} a_{ij}x_ix_j-\sum_{i=1}^{n}b_ix_i$$

由于 A 正定，所以在R^n内，二次函数$F(\cdot)$有唯一的极小点 $\boldsymbol{u}$.

利用 A 的对称性，二次函数$F(\cdot)$对 x_i 的偏导数为

7.13.3 $$\frac{\partial F}{\partial x_i}=\sum_{j=1}^{n} a_{ij}x_j-b_i\equiv -r_i$$

亦即$F(\cdot)$对 $\boldsymbol{x}$ 的斜量是

7.13.4 $\text{grad } F(\boldsymbol{x})=A\boldsymbol{x}-\boldsymbol{b}=-\boldsymbol{r}$

$F(\cdot)$的极小点 $\boldsymbol{u}$ 满足方程

7.13.5 $\text{grad } F(\boldsymbol{u})=0$, 即 $A\boldsymbol{u}=\boldsymbol{b}$

也就是说，二次函数 $F(\cdot)$ 的极小化问题和求解方程组 (7.13.1) 等价.因此，当 A 正定时，求解 (7.13.1) 的问题可转化为求二次函数$F(\cdot)$的极小值问题.

7.13.2 最速下降法

最速下降法(Steepest Descent Method)是求二次函数 $F(\cdot)$的极小点 $\boldsymbol{u}$ 的一种迭代方法.从初始猜测值$\boldsymbol{x}^{(0)}$出发，按

7.13.6 $\boldsymbol{x}^{(k+1)}=\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{P}^{(k)}$

迭代，其中 α_k 是使

7.13.7 $$F(\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{P}^{(k)})=\min_{\alpha} F(\boldsymbol{x}^{(k)}+\alpha\boldsymbol{P}^{(k)})$$

的迭代系数.由 (7.13.2) 有

$$F(\boldsymbol{x}^{(k)}+\alpha_k\boldsymbol{P}^{(k)})$$

$$=\frac{1}{2}[A(\boldsymbol{x}^{(k)}+\alpha\boldsymbol{P}^{(k)}),(\boldsymbol{x}^{(k)}+\alpha\boldsymbol{P}^{(k)})]-(\boldsymbol{b},\boldsymbol{x}^{(k)}+\alpha\boldsymbol{P}^{(k)})$$

$$=\frac{1}{2}A(\boldsymbol{x}^{(k)},\boldsymbol{x}^{(k)})+\frac{1}{2}\alpha(A\boldsymbol{x}^{(k)},\boldsymbol{P}^{(k)})+\frac{\alpha}{2}(A\boldsymbol{P}^{(k)},\boldsymbol{x}^{(k)})$$

$$+\frac{\alpha^2}{2}(AP^{(k)},P^{(k)})-(b,x^{(k)})-\alpha(b,P^{(k)})$$

$$=\frac{1}{2}\alpha^2(AP^{(k)},P^{(k)}-\alpha(r^{(k)}),P^{(k)})+F(x^{(k)})$$

所以从

$$\frac{dF(x^{(k)}+\alpha P^{(k)})}{d\alpha}$$

$$=\alpha(AP^{(k)},P^{(k)})-(r^{(k)},P^{(k)})=0$$

可以得出

$$\alpha_k=\alpha_{\min}=\frac{(r^{(k)},P^{(k)})}{(AP^{(k)},P^{(k)})} \tag{7.13.8}$$

(7.13.6)式中$P^{(k)}$是$x^{(k)}$的修正方向，为了使$F(x)$减小得最快，应取F（•）在$x^{(k)}$的负梯度方向，即

$$P^{(k)}=-\text{grad }F(x)=r^{(k)}=b-Ax^{(k)} \tag{7.13.9}$$

由（7.13.6），（7.13.8）和（7.13.9）构成的迭代法称为**最速下降法**．取任意初始向量$x^{(0)}$，有

$$\begin{cases} r^{(k)}=b-Ax^{(k)} \\ \alpha^{(k)}=\dfrac{(r^{(k)},r^{(k)})}{(Ar^{(k)},r^{(k)})} \\ x^{(k+1)}=x^{(k)}+\alpha_k r^{(k)} \end{cases} \quad (k=0,1,2,\cdots) \tag{7.13.10}$$

这是一种非线性迭代方法，只要 A 正定，它肯定收敛．然而，理论分析与实际计算均表明最速下降法收敛较慢．

7.13.3 共轭斜量法

共轭斜量法(Conjugate Gradient Method)简称CG法，是一种不需要选取任何迭代参数的非线性迭代方法，是按所谓共轭关系来选择修正方向的，它是被称作为共轭方向法的一类算法中最重要的一种方法．共轭斜量法适合于系数矩阵

一五

为对称正定的情况。如果没有舍入误差，理论上它能保证最多迭代n步（n为方程组的阶数）便求得$Ax=b$的精确解，因而实质上也是一种直接方法。其计算步骤如下：

第1步，从任意的初始向量$x^{(0)}=(x_1^{(0)},x_2^{(0)},\cdots x_n^{(0)})$出发，与最速下降法一样选取负梯度方向

7.13.11 $$\begin{cases} P^{(1)}=r^{(0)}=-\operatorname{grad}\quad F(x^{(0)}) \\ \alpha_1=\dfrac{(r^{(0)},P^{(1)})}{(AP^{(1)},P^{(1)})} \\ x^{(1)}=x^{(0)}+\alpha_1 r^{(0)} \end{cases}$$

第k步，修正方向$P^{(k)}$不取负梯度方向，而将$P^{(k)}$表示成$r^{(k-1)}$和$P^{(k-1)}$的线性组合

7.13.12 $$P^{(k)}=r^{(k-1)}+\eta_k P^{(k-1)}\qquad (k=2,3,\cdots)$$

且$P^{(k)}$是$P^{(k-1)}$的共轭方向，它们满足共轭关系

7.13.13 $$(AP^{(k)},P^{(k-1)})=(P^{(k)},AP^{(k-1)})=0$$

将（7.13.12）代入（7.13.13）可确定η_k

7.13.14 $$\eta_k=-\frac{(r^{(k-1)},AP^{(k-1)})}{(P^{(k-1)},AP^{(k-1)})}$$

按

7.13.15 $$x^{(k)}=x^{(k-1)}+\alpha_k P^{(k)}$$

式迭代，其中α_k使

$$F(x^{(k)})=F(x^{(k-1)}+\alpha_k P^{(k)})=\min_{\alpha} F(x^{(k-1)}+\alpha P^{(k)})$$

亦即，使二次函数$F(\cdot)$达到最小的迭代系数，可得

7.13.16 $$\alpha_k=\frac{(r^{(k-1)},P^{(k)})}{(AP^{(k)},P^{(k)})}$$

利用（7.13.11），（7.13.12），（7.13.14），（7.13.15）和（7.13.16），可以将共轭斜量法的计算公式归纳如下：

任取初始向量$x^{(0)}$，并计算

7.13.17
$$\begin{cases} r^{(0)}=b-Ax^{(0)} \\ P^{(1)}=r^{(0)} \\ \alpha_1=\dfrac{(r^{(0)},P^{(1)})}{(AP^{(1)},P^{(1)})} \\ x^{(1)}=x^{(0)}+\alpha_1 P^{(1)} \\ r^{(1)}=b-Ax^{(1)} \end{cases}$$

7.13.18
$$\begin{cases} \eta_{k-1}=-\dfrac{(r^{(k-1)},AP^{(k-1)})}{(P^{(k-1)},AP^{(k-1)})} \\ P^{(k)}=r^{(k-1)}+\eta_{k-1}P^{(k-1)} \\ \alpha_k=\dfrac{(r^{(k-1)},P^{(k)})}{(AP^{(k)},P^{(k)})} \qquad (k=2,3,\cdots) \\ x^{(k)}=x^{(k-1)}+\alpha_k P^{(k)} \\ r^{(k)}=b-Ax^{(k)} \end{cases}$$

作简化后，可得便于计算的公式

7.13.19
$$\begin{cases} \text{任取} x^{(0)} \\ r^{(0)}=b-Ax^{(0)} \\ P^{(1)}=r^{(0)} \end{cases}$$

7.13.20
$$\begin{cases} \alpha_k=\dfrac{(r^{(k-1)},r^{(k-1)})}{(AP^{(k)},P^{(k)})} \\ x^{(k)}=x^{(k-1)}+\alpha_k P^{(k)} \\ r^{(k)}=r^{(k-1)}-\alpha_k AP^{(k)} \qquad (k=1,2,\cdots) \\ \eta_k=\dfrac{(r^{(k)},r^{(k)})}{(r^{(k-1)},r^{(k-1)})} \\ P^{(k+1)}=r^{(k)}+\eta_k P^{(k)} \end{cases}$$

共轭斜量法最初是作为解线性代数方程组的一个方法提出来的，后来，证明可以直接把它推广到解非线性方程组和求函数极小值问题.共轭斜量法是求解系数矩阵为大型稀疏、对称正定的线性方程组的非常有效的方法.在整个计算过程中，A的形状保持不变，公式内只包含$AP^{(k)}$和一些向量的点

积.按照上述共轭斜量公式计算，最多计算 n 步（n 为方程组的阶数），便可得到方程组（7.13.1）的真解。

7.13.21 **例** 用共轭斜量法（7.13.19）和（7.13.20）解方程组 $Ax=b$，其中

$$A=\begin{pmatrix} 1 & -\frac{1}{2} & 0 \\ -\frac{1}{2} & 1 & -\frac{1}{2} \\ 0 & -\frac{1}{2} & 1 \end{pmatrix},\quad b=\begin{pmatrix} 0 \\ 0 \\ 2 \end{pmatrix}$$

解 取初值 $x^{(0)}=(2,\ -3,\ 2)^T$

1° $r^{(0)}=P^{(1)}=(-3.5,\ 5.0,\ -1.5)^T$

$AP^{(1)}=(-6.0,\ 7.5,\ -4.0)^T$

$$\alpha_1=\frac{(r^{(0)},r^{(0)})}{(AP^{(1)},P^{(1)})}=\frac{39.5}{64.5}=0.6124$$

$x^{(1)}=x^{(0)}+\alpha_1 P^{(1)}=(-0.1431,0.0620,1.0814)^T$

2° $r^{(1)}=(0.1744,0.4070,0.9500)^T$

$$\eta_1=\frac{(r^{(1)},r^{(1)})}{(r^{(0)},r^{(0)})}=\frac{1.0978}{39.5}=0.02779$$

$P^{(2)}=r^{(1)}+\eta_1 P^{(1)}=(0.0771,0.5459,0.9079)^T$

$AP^{(2)}=(-0.1958,0.0534,0.6350)^T$

$\alpha_2=1.8602$

$x^{(2)}=(0.0,1.0769,2.7692)^T$

3° $r^{(2)}=(0.5385,0.3077,-0.2308)^T$

$\eta_2=0.3989$

$P^{(3)}=(0.5692,0.5254,0.1314)^T$

$AP^{(3)}=(0.3065,0.1751,0.1314)^T$

$\alpha_3=1.7568$

$x^{(3)}=x^{(2)}+\alpha_3 P^{(3)}=(1.0000,2.0000,3.0000)^T$

$r^{(3)}=b-Ax^{(3)}=0$。

计算三步后可求出精确解。

7.14

7.14 共轭斜量加速方法

共轭斜量法不是单纯的一个方法，而是一类方法．其中每个都可看作是对于某个特定的一阶线性定常迭代法(7.10.1)的加速过程．古典CG法(7.13.17)和(7.13.18)或(7.13.19)和(7.13.20)可以看作是对RF法(7.5.10)的加速过程并表示成三项形式。当CG加速法应用于更一般的基本方法时，也可以导出类似于Чебышев（切比雪夫）加速过程的三项式（7.15.2)．对于同一个基本迭代法，当用同一种方法度量误差时，CG加速法至少与相应的Чебышев过程一样快。此外，在实现CG法加速过程中不需要作任何参数估计，因此，近几年来CG加速法得到了广泛的应用。

7.14.1 共轭斜量法的三项形式

设 A 是对称正定矩阵，考虑求解 n 阶线性代数方程组

7.14.1 $Ax=b$

古典CG法的三项形式是

7.14.2 $x^{(k+1)}=\omega_{k+1}(\xi_{k+1}r^{(k)}+x^{(k)})+(1-\omega_{k+1})x^{(k-1)}$

其中

$$\xi_{k+1}=\frac{(r^{(k)},r^{(k)})}{(r^{(k)},Ar^{(k)})}$$

$$\omega_1=1$$

$$\omega_{k+1}=\frac{1}{1-\dfrac{(r^{(k)},r^{(k)})\xi_{k+1}}{(r^{(k-1)},r^{(k-1)})\xi_k\omega_k}}\qquad(k\geqslant1)$$

$$r^{(k)}=b-Ax^{(k)}$$

7.14.2 共轭斜量加速法

将古典CG法的三项形式应用到一阶定常线性迭代法(7.10.1)

7.14.3 $x^{(k+1)}=Bx^{(k)}+f$

令伪残余向量（Pseudo-Residual Vector）

7.14.4 $\delta^{(k)}=Bx^{(k)}+f-x^{(k)}$

得到CG加速公式

7.14.5
$$x^{(k+1)}=\omega_{k+1}\{\xi_{k+1}(Bx^{(k)}+f)+(1-\xi_{k+1})x^{(k)}\}+(1-\omega_{k+1})x^{(k-1)}$$
$$=\omega_{k+1}\{\xi_{k+1}\delta^{(k)}+x^{(k)}\}+(1-\omega_{k+1})x^{(k-1)}$$

其中加速参数序列ξ_k和ω_k的计算公式为：

7.14.6
$$\xi_{k+1}=\frac{1}{1-\dfrac{\delta^{(k)T}W^TWB\delta^{(k)}}{\delta^{(k)T}W^TW\delta^{(k)}}}$$

$\omega_1=1$

7.14.7
$$\omega_{k+1}=\frac{1}{1-\dfrac{\delta^{(k)T}W^TW\delta^{(k)}\xi_{k+1}}{\delta^{(k-1)T}W^TW\delta^{(k-1)}\xi_k\omega_k}}$$

式中W是使$W(I_n-B)W^{-1}$为对称正定矩阵的可对称化矩阵。适于用CG法加速的基本迭代方法的迭代矩阵B有

7.14.8
$$B=\begin{cases} I_n-A, & \text{RF 迭代法}\\ I_n-D^{-1}A, & \text{Jacobi 迭代法}\\ (D-\omega U)^{-1}[\omega L+(1-\omega)D](D-\omega L)^{-1}[\omega U+(1-\omega)D], & \text{SSOR迭代法}\end{cases}$$

其中

$A=D-L-U$

D，L，U分别是A的对角部分，严格下三角部分和严格上三角部分。

在所有多项式加速过程中，CG加速过程使误差$\varepsilon^{(k)}$的

$[W^TW(I_n-B)]^{\frac{1}{2}}$范数达到极小值，若令$\widehat{\varepsilon^{(k)}}=W\varepsilon^{(k)}$，亦即使$(\varepsilon^{(k)}, W^TW(I_n-B)\varepsilon^{(k)})$ 达到极小值。

CG加速方法有以下性质：

1° 如果基本迭代是RF迭代（或Richardson（李查逊）迭代），那么

$B=I_n-A,\ \delta^{(k)}=r^{(k)}=b-Ax^{(k)},\ W=I_n$

则CG加速法就是古典CG法的三项形式。

2° CG加速法需要对称化矩阵W，更确切地说需要W^TW。取

$$W=\begin{cases} I_n\ , & \text{RF迭代法} \\ D^{\frac{1}{2}}\ , & \text{Jacobi迭代法} \\ \dfrac{1}{\sqrt{\omega(2-\omega)}}D^{-\frac{1}{2}}(D+\omega U), & \text{SSOR迭代法} \end{cases}$$

后，其伪残余量是W正交的，即

$$(W\delta^{(i)}, W\delta^{(j)})=0 \qquad (i\neq j)$$

3° 计算参数ξ_{k+1}时需要计算$B\delta^{(k)}$，为避免计算(7.14.5)式中的$Bx^{(k)}$，则利用公式

7.14.9 $$\delta^{(k+1)}=\omega_{k+1}\{\xi_{k+1}B\delta^{(k)}+(1-\xi_{k+1})\delta^{(k)}\}+(1-\omega_{k+1})\delta^{(k-1)}$$

计算伪残余量。

7.15

7.15 Чебышев加速法

用Чебышев多项式构成一种半迭代法，能使一阶线性定常迭代过程(7.10.1)得到加速，这样的半迭代法称为Чебышев半迭代法(Semi-iterative Method)。

考虑一阶线性定常迭代法

$x^{(k+1)}=Bx^{(k)}+f$

其中迭代矩阵B的特征值λ是实的，并满足不等式

7.15.1 $m\leqslant\lambda\leqslant M<1$

$m=\lambda_{\min}(B)$和 $M=\lambda_{\max}(B)$ 分别表示 B 的特征值 λ 的最小值和最大值。

Чебышев 半迭代公式为

7.15.2 $x^{(k+1)}$

$=\omega_{k+1}\{\xi(Bx^{(k)}+f)+(1-\xi)x^{(k)}\}+(1-\omega_{k+1})x^{(k-1)}$

其中 ξ 是常数，ω_k是迭代参数，其计算公式

7.15.3 $\xi=\dfrac{2}{2-M-m}$

7.15.4 $\sigma=\dfrac{M-m}{2-M-m}$

7.15.5
$$\begin{cases}\omega_1=1\\ \omega_2=\dfrac{1}{1-\dfrac{1}{2}\sigma^2}\\ \omega_{k+1}=\dfrac{1}{1-\dfrac{1}{4}\sigma^2\omega_k} & (k\geqslant 2)\end{cases}$$

从任意初始值$x^{(0)}$开始迭代，一直进行到

7.15.6 $\dfrac{1}{1-M}\dfrac{\|\delta^{(k)}\|_2}{\|x^{(k)}\|_2}<\varepsilon$

为止。其中 ε 为预先给定的迭代精度，例如 $\varepsilon=10^{-6}$；$\delta^{(k)}$是**伪残余向量** (Pseudo-Residual Vector)

7.15.7 $\delta^{(k)}=Bx^{(k)}+f-x^{(k)}$

适于用 Чебышев加速法加速的基本迭代法的迭代矩阵B有

$$B=\begin{cases}I_n-D^{-1}A, & \text{Jacobi迭代法}\\ (D-\omega U)^{-1}[\omega L+(1-\omega)D](D\\ \quad-\omega L)^{-1}[\omega U+(1-\omega)D], & \text{SSOR迭代法}\end{cases}$$

其中 A 是方程组 $Ax=b$ 的系数矩阵，且 $A=D-L-U$。

如果（7.15.3），（7.15.4）和（7.15.6）中的 m 和 M 是精确值，那么Чебышев加速法比 Jacobi 迭代法和 SSOR 迭代法的计算快得多。但是，由于难以知道它们的精确值，故只能采用估计值 m_E 和 M_E。其中 M_E 是最重要的，因为Чебышев 加速法的收敛速度对 m_E 的误差不很敏感，特别当 $m_E<m$ 时更是如此；但对 M_E 的误差却非常敏感，尤其当 M_E 小于 M 且接近于 M 时，则收敛速度有显著地提高。在加速过程中用自适应方法可得到较精确的估值 M_E。在 Чебышев 加速 Jacobi 迭代时，取 $m_E=-M_E$；在Чебышев 加速 SSOR 迭代时，取 $m_E=0$，同时自适应地求出 SSOR 的好的松弛参数 ω^*。

当矩阵 A 对称正定且具有性质 'A'（定义 7.3.18）和相容次序（定义7.3.24）时，从渐近收敛速度来说，具有最优松弛因子 ω_{opt}（7.8.2）的 SOR 迭代法的收敛速度是 Чебышев 加速法加速 Jacobi（雅可比）迭代的 1.5 倍到 2 倍。如果矩阵 A 是对称正定阵，但不具有性质 'A' 和非负等特性，则一般说来 SOR 法收敛较慢，此时，采用Чебышев 加速Jacobi 迭代可能取得较好的效果，但存储量增加。

7.15.8 例 假设矩阵 A 是对称正定的，列出用 Jacobi 迭代的Чебышев加速法求解方程组

$$Ax=b$$

的计算公式。

解 Jacobi 迭代公式是

$$x^{(k+1)}=B_Jx^{(k)}+f$$

其中 $B_J=I_n-D^{-1}A$，D 是 A 的对角线元素组成的正定的对角线矩阵，$f=D^{-1}b$。存在可对称化矩阵 $W=D^{\frac{1}{2}}$，使 $\widetilde{B}_J=D^{\frac{1}{2}}B_JD^{-\frac{1}{2}}=I_n-D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ 为对称矩阵，$\widetilde{B}_J$ 的特征

值为实数，所以 B_J 的特征值 $\lambda(B_J)$ 均为实数。同时，由于 $D^{-\frac{1}{2}}AD^{-\frac{1}{2}}$ 是正定矩阵，B_J 的特征值 $\lambda(B_J)$ 还应该是小于 1 的实数，满足不等式（7.15.1）。令（7.15.2）式中的 $B=B_J$，$\boldsymbol{f}=\boldsymbol{f}_J$，则 Jacobi 迭代的Чебышев 半迭代公式可表示为

7.15.9 $$\boldsymbol{x}^{(k+1)}=\omega_{(k+1)}\{\xi(B_J\boldsymbol{x}^{(k)}+\boldsymbol{f}_J)+(1-\xi)\boldsymbol{x}^{(k)}\}+(1-\omega_{k+1})\boldsymbol{x}^{(k-1)}$$

其中各参数按（7.15.3），（7.15.4）和（7.15.5）计算。

如果矩阵 A 是形如(7.3.16)的具有相容次序和性质'A'的矩阵，可以证明 $\rho(B_J)<1$，故可以令 $M_B=-m_B=\rho(B_J)$。因此，由（7.15.3）得参数 $\xi=1$，$\sigma=\rho(B_J)$，ЧебЫшев半迭代的计算公式可写为

7.15.10 $$\boldsymbol{x}^{(k+1)}=\omega_{k+1}(B_J\boldsymbol{x}^{(k)}+\boldsymbol{f}_J)+(1-\omega_{k+1})\boldsymbol{x}^{(k-1)}$$

$$\omega_1=1$$

$$\omega_2=\frac{2}{2-\rho(B_J)^2}$$

$$\omega_{k+1}=\frac{1}{1-\frac{1}{4}\rho(B_J)^2\omega_k}\qquad(k\geqslant 2)$$

7.16

7.16 Lanczos加速法

考虑用基本迭代法

7.16.1 $$\boldsymbol{x}^{(k+1)}=B\boldsymbol{x}^{(k)}+\boldsymbol{f}$$

求解方程组

$$A\boldsymbol{x}=\boldsymbol{b}$$

7.16.2 **定义** 如果存在某个非奇异矩阵 P，使得 $P(I_n-B)P^{-1}$ 为对称正定矩阵，则称基本迭代法（7.16.1）是**可对称化的**。若不存在这样的矩阵 P，则基本迭代法（7.16.1）是**不可对称化的**。

Lanczos（兰错斯）加速法是加速不可对称化基本迭代法的，任意给定初值$\boldsymbol{x}^{(0)}$和$\boldsymbol{R}^{(0)}$。

7.16.3 Lanczos 加速公式为

$$\boldsymbol{r}^{(0)}=\boldsymbol{b}-A\boldsymbol{x}^{(0)}$$

$$\boldsymbol{x}^{(k+1)}=\omega_{k+1}(\xi_{k+1}\boldsymbol{r}^{(k)}+\boldsymbol{x}^{(k)})+(1-\omega_{k+1})\boldsymbol{x}^{(k-1)}$$

$$\boldsymbol{r}^{(k+1)}=-\omega_{k+1}\xi_{k+1}A\boldsymbol{r}^{(k)}+\omega_{k+1}\boldsymbol{r}^{(k)}+(1-\omega_{k+1})\boldsymbol{r}^{(k-1)}$$

$$\boldsymbol{R}^{(k+1)}=-\omega_{k+1}\xi_{k+1}A^T\boldsymbol{R}^{(k)}+\omega_{k+1}\boldsymbol{R}^{(k)}+(1-\omega_{k+1})\boldsymbol{R}^{(k-1)}$$

$$\xi_{k+1}=\frac{(\boldsymbol{r}^{(k)},\boldsymbol{R}^{(k)})}{(A\boldsymbol{r}^{(k)},\boldsymbol{R}^{(k)})}$$

$$\omega_1=1$$

$$\omega_{k+1}=\frac{1}{1-\dfrac{\xi_{k+1}}{\xi_k\omega_k}\,\dfrac{(\boldsymbol{r}^{(k)},\boldsymbol{R}^{(k)})}{(\boldsymbol{r}^{(k-1)},\boldsymbol{R}^{(k-1)})}}\qquad(k\geqslant1)$$

其中

$$\boldsymbol{R}^{(k)}=\widetilde{\boldsymbol{b}}-\widetilde{A}^T\boldsymbol{x}^{(k)}$$

是辅助方程

$$\widetilde{A}^T\widetilde{\boldsymbol{x}}^{(k)}=\widetilde{\boldsymbol{b}}$$

的残余向量。关于$\widetilde{\boldsymbol{b}}$的选取在上述公式中已经避免。

可以把以上的 Lanczos 过程（7.16.3）看成是对 RF 法的加速过程，在不可对称化的情况下，一旦计算中某一步$\boldsymbol{r}^{(k)}\neq0$而（$\boldsymbol{r}^{(k)}$，$\boldsymbol{R}^{(k)}$）$=0$，例如$\boldsymbol{R}^{(0)}$选得不好，则 Lanczos法可能失败。如果($\boldsymbol{r}^{(0)}$，$\boldsymbol{R}^{(0)}$)，($\boldsymbol{r}^{(1)}$，$\boldsymbol{R}^{(1)}$)，…都不为零，那么最多用 n 步（方程组阶数）Lanczoc 方法就可收敛。

现考虑用 Lanczos 法去加速基本迭代法（7.16.1）的收敛，则（7.16.1）可改写成

7.16.4 $(I_n-B)\boldsymbol{x}=\boldsymbol{f}$

把（7.16.4）看作是$A\boldsymbol{x}=\boldsymbol{b}$的一种预处理，其中 B 是基本迭代法的迭代矩阵，例如 Jacobi（雅可比）迭代矩阵 $B=B_J=I_n-D^{-1}A$ 等。在 Lanczos 法中，如果用 $\delta^{(k)}=B\boldsymbol{x}^{(k)}+\boldsymbol{f}-$

$x^{(k)}$代替 $r^{(k)}$，用 I_n-B 代替 A，令 $s^{(k)}$ 表示$(I_n-B)^T\widetilde{x}=\widetilde{f}$的残余向量，$\widetilde{f}$ 的选取问题与 $\widetilde{b}$ 一样已避免，用 $s^{(k)}$代替$R^{(k)}$，则可得到更一般的方法。

7.16.5 Lanczos 加速公式

任意给定向量 $x^{(0)}$ 和$s^{(0)}$

$$\delta^{(0)}=Bx^{(0)}+f-x^{(0)}$$

$$x^{(k+1)}=\omega_{k+1}(\xi_{k+1}\delta^{(k)}+x^{(k)})+(1-\omega_{k+1})x^{(k-1)}$$

$$\delta^{(k+1)}=-\omega^{k+1}\xi_{k+1}(I_n-B)\delta^{(k)}+\omega_{k+1}\delta^{(k)}+(1-\omega_{k+1})\delta^{(k-1)}$$

$$s^{(k+1)}=-\omega_{k+1}\xi_{k+1}(I_n-B)^Ts^{(k)}+\omega_{k+1}s^{(k)}+(1-\omega_{k+1})s^{(k-1)}$$

$$\xi_{k+1}=\frac{(\delta^{(k)},s^{(k)})}{((I_n-B)\delta^{(k)},s^{(k)})}$$

$$\omega_1=1,$$

$$\omega_{k+1}=\frac{1}{1-\frac{\xi_{k+1}}{\xi_k}\frac{(\delta^{(k)},s^{(k)})}{(\delta^{(k-1)},s^{(k-1)})}\frac{1}{\omega_k}}\qquad(k\geqslant1)$$

这里略去了 Lanczos 方法的构造过程，仅叙述方法的计算公式。

7.17

7.17 GCW法的加速过程

7.17.1 GCW 法

求解 n 阶方程组

7.17.1 $Ax=b$

其中系数矩阵 A 是正实的（positive real），即 $A+A^T$ 是对称正定的（定义7.3.10）。令

7.17.2 $Q=\frac{1}{2}(A^T+A)$

则基本迭代公式为

7.17.3 $$x^{(k+1)}=Q^{-1}\frac{1}{2}(A^T-A)x^{(k)}+Q^{-1}b$$
$$=Bx^{(k)}+f$$

其中

7.17.4 $$B=Q^{-1}\frac{1}{2}(A^T-A)$$
$$=Q^{-1}(Q-A)=I-Q^{-1}A=Q^{-1}R$$

7.17.5 $f=Q^{-1}b$

7.17.6 $R=Q-A$

迭代法（7.17.8）是Concus（康卡斯），Golub（哥拉布）（1976）和Widlund（怀德伦德）（1978）研究过的，称为GCW方法。为了进行GCW法的一次迭代，必须求解一个辅助方程组

7.17.7 $Qx^{(k+1)}=Rx^{(k)}+b$

其系数矩阵是对称正定的，往往可用直接方法求解，也可用迭代法求解。

GCW方法是不可对称化的（7.16.2），但有一个重要性质——迭代矩阵B的特征值都是纯虚数。在这种情况下，可以利用Чебышев（切比雪夫）加速法和共轭斜量（CG）加速法。

7.17.2 迭代矩阵B的特征值为纯虚数时的Чебышев加速法

Чебышев加速法能有效地加速GCW方法。因为GCW法中迭代矩阵B的特征值是纯虚数，相应的Чебышев加速公式是

7.17.8 $x^{(k+1)}=\omega_{k+1}(Bx^{(k)}+f)+(1-\omega_{k+1})x^{(k-1)}$

7.17.9 $\omega_1=1$

7.17.10 $$\omega_2=\frac{1}{1+\frac{1}{2}b^2}$$

7.17.11 $\omega_{k+1}=\dfrac{1}{1+\dfrac{1}{4}b^2\omega_k}$ $(k\geq 2)$

取$b=\rho(B)$，$\rho(B)$是矩阵B的谱半径（定义7.2.17），(7.17.8) 中的Q，B，f，R由（7.17.2），（7.17.4）-（7.17.6）等式给出。

7.17.3 基于GCW法的广义CG加速过程

Concus，Golub 和 Widlund 发展了一个基于 GCW法的广义 CG 加速过程，其方法如下：

7.17.12 $x^{(k+1)}=\omega_{n+1}(Bx^{(k)}+f)+(1-\omega_{n+1})x^{(k-1)}$

7.17.13 $\omega_1=1$

7.17.14 $\omega_{k+1}=\left[1+\dfrac{(Q^{\frac{1}{2}}\delta^{(k)},Q^{\frac{1}{2}}\delta^{(k)})}{(Q^{\frac{1}{2}}\delta^{(k-1)},Q^{\frac{1}{2}}\delta^{(k-1)})}\dfrac{1}{\omega_k}\right]^{-1}$ $(k\geq 1)$

其中

7.17.15 $\delta^{(k)}=Bx^{(k)}+f-x^{(k)}$

由（7.17.2），（7.17.4）-（7.17.6）给出（7.17.12）中的Q、B、f和R。

7.18

7.18 多重网格法

多重网格法（Multiple Grid Method）简称MG方法，它是人们早已熟知的二种数值方法的有机结合，即用有效地光滑误差的松弛法，对细网上的近似值作粗网修正。正是这种结合使多重网格法成为（渐近）最优的（Asymptotically Optimal）迭代方法。如果把MG方法与套迭代（Nested Iteration）相结合，则效果更加显著，达到预定精确度所需要的计算工作量与未知数的个数N成正比，即$O(N)$，此外，多重网格的收敛因子与h（网格步长）无关。

1977年以来，多重网格法已广泛地用于求解椭圆型边值问题，同时向数值数学的许多方面渗透。例如，用多重网格法求解抛物型问题以及其它不定常问题和非椭圆型问题、本征值问题、非线性问题等。此外，多重网格法亦可有效地用于求解积分方程。在向量机上或在并行计算机上使用多网格方法更加有效，所以多重网格法是一种通用的迭代方法。

7.18.1 多重网格方法的基本原理

要求解的连续问题是

7.18.1 $Lu=f$

其中 L 为微分算子或积分算子或泛函求极值算子。例如，二维 poisson（泊松）方程第一边值问题

7.18.2 $Lu=f(x,y), \qquad (x,y)\in\Omega$

7.18.3 $u=g(x,y), \qquad (x,y)\in\partial\Omega$

其中$L=-\Delta=-(\frac{\partial^2}{\partial x^2}+\frac{\partial^2}{\partial y^2})$ 为微分算子，Ω 表示解域，$\partial\Omega$ 表示 Ω 的边界。

不论用有限差分法还是用有限元法求解上述问题，首先将平面域进行剖分。若用有限差分法，则作 x 轴和 y 轴的两族平行线，将 Ω 剖分成矩形或正方形子域，设 $\Delta x=\Delta y=h$，将这种剖分得到的网格点的集合称为 Ω_h。若用有限元方法，则将 Ω 作三角形或矩形剖分，所得的网格点的集合亦称为 Ω_h。

在 Ω_h 内的每一个网格点上可建立微分方程（7.18.2）相应的差分方程，用 u_{ij} 表示 $u(x_i, y_j)$ 的近似值，用五点差分格式得到的差分方程为

7.18.4 $$-\Delta_h U_{ij}=-\frac{U_{i+1,j}+U_{i-1,j}+U_{i,j+1}+U_{i,j-1}-4U_{ij}}{h^2}$$

$$=f_{ij}, \qquad (x_i,y_j)\in\Omega_h$$

7.18.5　$U_{ij}=g_{ij}$　　$(x_i,y_j)\in\partial\Omega_h$

其中 $\partial\Omega_h$为边界上网格点的集合。

用矩阵形式表示（7.18.4）和（7.18.5），得

7.18.6　$L_h\boldsymbol{u}_h=\boldsymbol{f}_h$

其中 L_h为 N 阶大型稀疏矩阵，$\boldsymbol{u}_h$和 $\boldsymbol{f}_h$为具有 N 个分量的列向量。对于（7.18.6），传统的迭代解法是在确定的一种网格 Ω_h上采用某种迭代解法，例如 Gauss-Seidel（高斯-赛得尔）法（7.6），SOR（逐次超松弛）法（7.7），ADI（交替方向隐式）法（7.11.4）以及它们的改进迭代技术。而多重网格方法恰恰不是在一种确定的网格上求解，它采用不同等级的网格，有一组网格 $\Omega_{h_0},\Omega_{h_1}$，$\Omega_{h_2},\cdots$，$\Omega_{h_M}$，称它们为**网格序列** Ω_{h_k}（$k=0$，1，2，…，M），简写为 $\Omega_k(k=0,1,\cdots,M)$。随着 k 的增加，差分网格越来越细，这些网格都用同一种方式剖分，都逼近同一个域 Ω。相应地有网格步长序列 $h_k(k=0,1,\cdots,M)$ 及差分算子序列 $L_k(R=0,1,\cdots,M)$。

在 Ω_k上求解

$L_k\boldsymbol{u}_k=\boldsymbol{f}_k$

的问题称为 Ω_k问题。

7.18.7　在 Ω_M上求解

$L_M\boldsymbol{u}_M=\boldsymbol{f}_M$

的问题称为 Ω_M问题。

MG 方法的基本思想是用较粗网格 $\Omega_k(k<M)$上的问题作为 Ω_M问题的近似，也就是说，首先近似地求解 Ω_k问题，因为 Ω_k 问题的代数方程组比 Ω_M 问题的代数方程组要少得多，解起来也容易得多。然后将 Ω_k 上的函数值转移到 Ω_M上，作为迭代解 Ω_M问题的初始近似值。同样，在求解 Ω_k 问题时，用比 Ω_k更粗一级的网格上的近似解作为 Ω_k 问题的迭代初值。

对于线性问题，令$\overline{u}_M$是（7.18.7）的近似解，u_M为其精确解，则有

7.18.8 $u_M = \overline{u}_M + v_M$

其中v_M为修正量．可以得到

7.18.9 $L_M \overline{u}_M = f_M - d_M$

将（7.18.8）代入上式，有

$L_M u_M - L_M v_M = f_M - d_M$

因为

$L_M u_M = f_M$

所以有

7.18.10 $L_M v_M = d_M$

其中

$d_M = f_M - L_M \overline{u}_M$

d_M即为亏损量（Defect），（7.18.10）被称为亏损方程．线性MG方法是把在细网Ω_M上迭代求解（7.18.7）与在粗网Ω_k上求解亏损方程

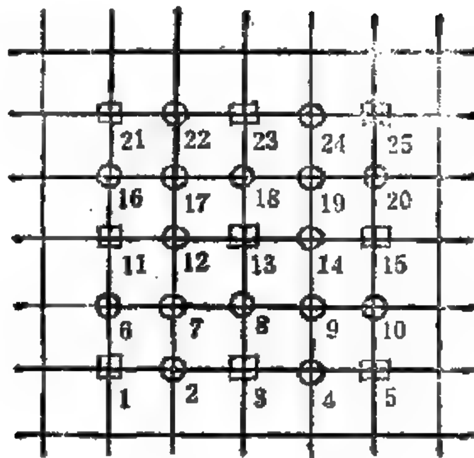
7.18.11 $L_k v_k = d_k$

相结合的一种方法，而求解亏损方程（7.18.11）本身又是把在Ω_k上迭代求解（7.18.11）与求解更粗网格上的亏损方程相结合．这就是线性MG方法的基本原理．

7.18.2 双网格方法

双网格方法采用二种网格，一种是粗网用Ω_H(或Ω_{h0})表示，另一种是细网用Ω_h（或Ω_{h_1}）表示．网格步长分别为H和h，一般取$H=2h$，在Ω_h和Ω_H上的差分算子分别为L_h和L_H并设其逆L_h^{-1}和L_H^{-1}存在．粗网点用□表示，细网点用○表示，当$H=2h$时，粗细网格点分布如图(7.18.12)所示．

7.18.12 图 双网格剖分



粗细二种网格上的值需要进行转移。

I_h^H称为**限制算子**（Restriction Operator），它表示由细网到粗网上值的转移，可以用各种方式进行。

7.18.13 例 $I_h^H=[1]_h^H$**称为直接映射算子**（Straight Injection Operator）。

7.18.14 $(I_h^H u_h)(x,y)=u_h(x,y),\quad (x,y)\in\Omega_h$

其中$u_h(x,y)$为细网上的函数值。实际上，在粗细网点重合处的节点粗网值即为该点的细网值。

7.18.15 例 $I_h^H \triangleq \frac{1}{16}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$ **称为完全加权算子**（Full Weighting Operator）。比如标号 13 的粗网点上的函数值按此定义为

$$u_H^{13}=\frac{1}{16}(4u_h^{13}+2u_h^{14}+2u_h^{18}+2u_h^{12}+2u_h^{8}+u_h^{19}$$

$$+u_h^{17}+u_h^{7}+u_h^{9})$$

记号$\triangleq$表示定义的意思。I_H^h称为**插 值 算 子**（Interpolation Operator），它表示粗网到细网上值的转移。

7.18.16 例 用双线性插值

$$I_H^h \triangleq \frac{1}{4}\begin{pmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{pmatrix}$$

表示粗网点上的值按此权系数分配到邻近细网点上．比如

$$u_h^7 = \frac{1}{4}(u_H^3 + u_H^{13} + u_H^{11} + u_H^1)$$

$$u_h^8 = \frac{1}{2}(u_H^3 + u_H^{13})$$

$$u_h^{12} = \frac{1}{2}(u_H^{11} + u_H^{13})$$

$$u_h^6 = \frac{1}{2}(u_H^{11} + u_H^1)$$

$$u_h^2 = \frac{1}{2}(u_H^1 + u_H^3)$$

$u_h^{13} = u_H^{13}$， $u_h^{11} = u_H^{11}$， $u_h^1 = u_H^1, u_h^3 = u_H^3$

用双网格方法迭代解

7.18.17 $L_h u_h = f_h$

由已知 $u_h^{(n)}$ 计算 $u_h^{(n+1)}$ 的一个迭代步或者一个循环（n 为迭代标号），共分三个阶段：

1° 以 $u_h^{(n)}$ 为初值，在细网 Ω_h 上对(7.18.17) 式作 ν_1 次迭代(后面称松弛)．一般 $\nu_1=1$ 或 2，得近似值 $\overline{u}_h^{(n)}$，这一过程和结果用

$\overline{u}_h^{(n)} := \text{Relax}^{\nu_1}(u_h^{(n)}, L_h, f_h)$

表示，其中记号:= 表示定义并赋值的意思．

2° 粗网修正

（1）计算细网亏损量

$d_h^{(n)} = f_h - L_h \overline{u}_h^{(n)}$

（2）限制亏损量（从细网到粗网转移亏损量）

$d_H^{(n)}=I_h^H d_h^{(n)}$

（3）在粗网 Ω_H上求

$L_H v_H^{(n)}=d_H^{(n)}$

的精确解 $v_H^{(n)}$

（4）插值（从粗网到细网转移修正量）

$v_h^{(n)}=I_H^h v_H^{(n)}$

（5）修正Ω_h上的节点函数值

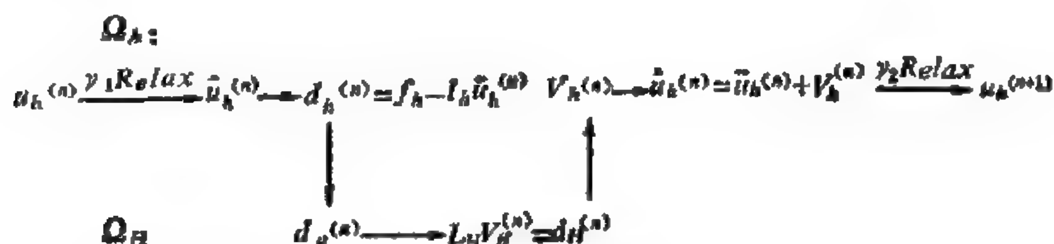
$\hat{u}_h^{(n)}=\bar{u}_h^{(n)}+v_h^{(n)}$

3° 以$\hat{u}_h^{(n)}$为初始近似值，在细网 Ω_h上对（7.18.17）式作ν_2次松弛。一般 $\nu_2=1$ 或 2，松弛结果

$u_h^{(n+1)}:=\text{Relax}^{\nu_2}(\hat{u}_h^{(n)},L_h,f_h)$

为直观起见，用（7.18.19）所示路径表示双网格法一个循环步的计算过程：

7.18.18



用符号表示，如图（7.18.19）所示。

7.18.19 图 线性（h,H）双网格方法

其中○表示在细网上松弛；

□表示在粗网上求精确解；

↘表示限制亏损量；↗表示插值并对细网上的解作修正。

上述过程的算子表达式为

$u_h^{(n+1)}=M_h^H u_h^{(n)}+G_h[f_h]$

其中$M_h^H=S_h^{\nu_2}\cdot(I_h-I_H^h L_H^{-1} I_h^H L_h)\cdot S_h^{\nu_1}$ 为双网格迭代算子；

$G_h[f_h]=(I_h-M_h^H)L_h^{-1}f_h$；$I_h$为单位算子；$S_h^{\nu_1}$和$S_h^{\nu_2}$为在细网$\Omega_h$上的迭代算子。

当$\nu_2=0$，$\nu_1\to\infty$时，双网格方法对一般问题是收敛的且与 h取何值无关。

双网格方法由二个要素组成：其一是在细网上松弛，其二是粗网修正。在细网上松弛，比如 Jacobi-ω 松弛即Jacobi（雅可比）外推松弛（7.10.2)，对于椭圆型边值问题能起到光滑误差的作用。例如，在单位正方域上第一边值条件的 Poisson（泊松）方程建立五点差分格式，即（7.18.4）的算子形式为（7.18.6)，x 和 y 二个方向都用等步长 h，$N=\frac{1}{h}$，可求出 L_h的特征函数和特征值，它们分别为

7.18.20 $\psi_{m_1,m_2}=2\sin(m_1\pi x)\ \sin(m_2\pi y)$

$$(x,y)\in\Omega_h,\max(m_1,m_2)\leqslant N-1$$

7.18.21 $\lambda_{m_1,m_2}=1-\frac{1}{2}(\cos(m_1\pi h)+\cos(m_2\pi h))$

而 Jacobi-ω 松弛公式是

$$\overline{u_h^{(n)}}=u_h^{(n)}+\omega(z_h-u_h^{(n)})$$

$$\frac{4}{h^2}z_h(x,y)+(L_hu_h^{(n)}(x,y)-\frac{4}{h^2}u_h^{(n)}(x,y)$$

$$=f_h(x,y),\qquad(x,y)\in\Omega_h$$

因此，Jacobi-ω 迭代矩阵是

7.18.22 $S_h=S_h(\omega)=I_h-\frac{\omega h^2}{4}L_h$

容易推出 S_h的特征函数和特征值，它们分别为

7.18.23 $\varphi_{m_1,m_2}=2\sin(m_1\pi x)\sin(m_2\pi y)$，$(x,y)\in\Omega_h$，

$$\max(m_1,m_2)\leqslant N-1$$

7.18.24 $\chi_{m_1,m_2}=\chi_{m_1,m_2}(\omega)$

$$=1-\frac{\omega h^2}{8}(2-\cos(m_1\pi h)-\cos(m_2\pi h))$$

把松弛前的误差

7.18.25 $v_h = u_h - u_h^{(n)}$

和松弛后的误差

7.18.26 $\overline{v}_h = u_h - \overline{u}_h^{(n)}$

展成 S_h 特征函数的离散级数，有

7.18.27 $$v_h = \sum_{\max(m_1, m_2) \leqslant N-1} \alpha_{m_1, m_2} \varphi_{m_1, m_2}$$

7.18.28 $$\overline{v}_h = \sum_{\max(m_1, m_2) \leqslant N-1} \chi_{m_1, m_2} \alpha_{m_1, m_2} \varphi_{m_1, m_2}$$

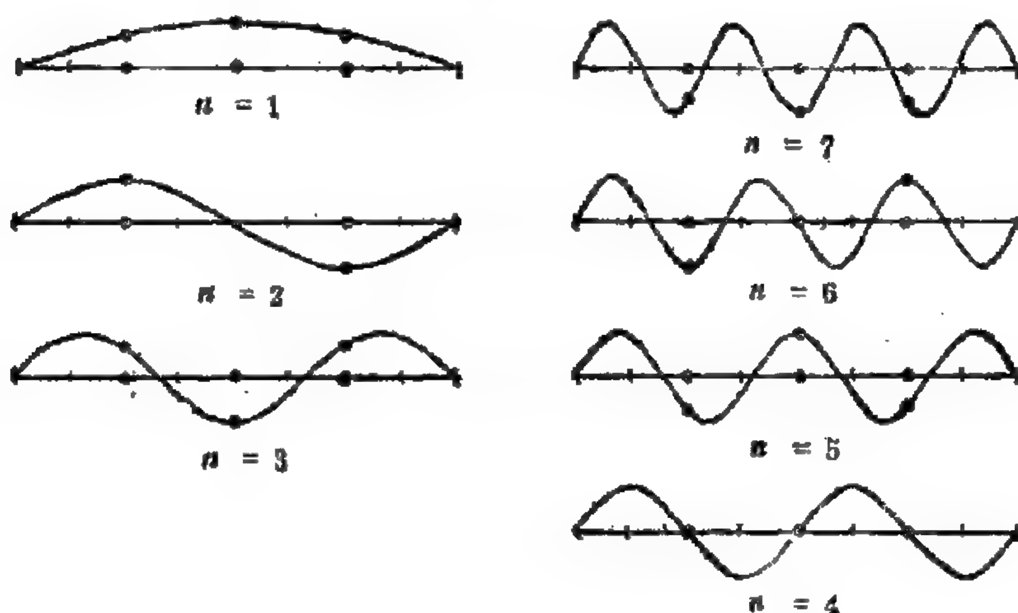
把以上级数中的各项分成高频和低频二部分。低频是指在粗网 Ω_H 上能表现出来的那部分特征函数；而高频是指在 Ω_H 上根本看不见的那部分特征函数，具体定义如下：

7.18.29 低频：φ_{m_1, m_2}，　$\max(m_1, m_2) < N/2$

7.18.30 高频：φ_{m_1, m_2}，$N/2 \leqslant \max(m_1, m_2) \leqslant N-1$

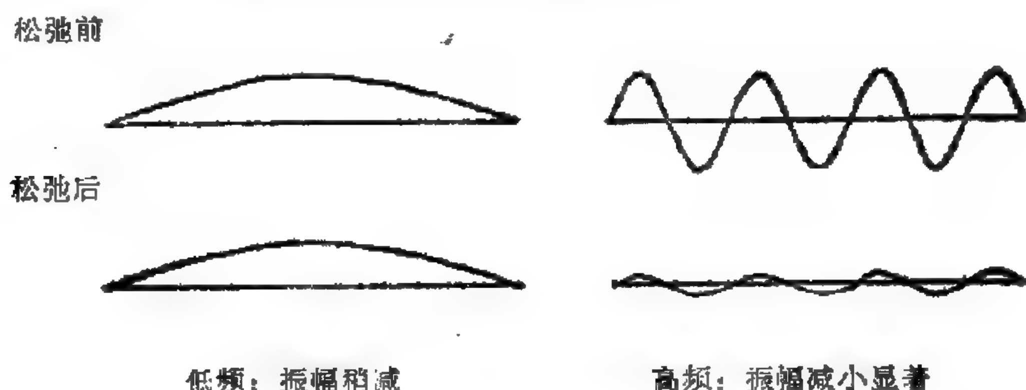
图（7.18.31）表示高频分量和低频分量。

7.18.31 图 $h = \frac{1}{8}$ 和 $H = \frac{1}{4}$ 时，$\sin(m\pi x)$ 的低频分量（$m=1,2,3$）和高频分量（$m=4,5,6,7$）。



图（7.18.32）表示二种频率分量在松弛前后其振幅的变化。

7.18.32 图 松弛方法典型的光滑误差特性



所以，在细网上作松弛能使误差的高频分量迅速衰减，而低频分量则减少得相当缓慢，使误差的光滑效应好但收敛很慢。

最光滑的特征函数$\varphi_{1,1}$是造成 Jacobi方法收敛慢的主要原因。另一方面，在粗网上由于未知量少，计算工作量比细网上小得多，高频分量显示不出来，低频分量恰很容易近似。因此，细网松弛和粗网修正通过限制和插值结合起来组成的双网格方法，把在细网上松弛使误差光滑的优点和在粗网上低频分量容易收敛的优点结合起来。双网格方法在实际计算中用得不多，但它是多重网格方法的理论基础。

7.18.3 线性多重网格方法

MG方法是在越来越粗的网格上递归地使用双网格方法，而在最粗的网格上精确地求解差分方程。MG 方法所用的符号和算子如下：

网格步长序列：$h_k(k=0,1,2,\cdots M)$，通常

$$h_k=\frac{h_{k-1}}{2}=2^{-k}h_0$$

网格序列：$\Omega_k(k=0,\ 1,\ 2,\ \cdots,\ M)$

差分算子序列：$L_k(k=0,\ 1,\ 2,\ \cdots M)$，设$L_k^{-1}$存在

松弛算子序列：Relax(u_k, L_k, f_k)

限制算子序列：$I_k^{k-1}(k=1, 2, \cdots M)$

插值算子序列：$I_{k-1}^k(k=1, 2, \cdots, M)$

其中 M 是网格重数。

用多重网格方法迭代求解方程组

7.18.33 $L_M u_M = f_M$

的计算过程是：已知 $u_M^{(n)}$求 $u_M^{(n+1)}$，仿双网格方法的三个阶段

1° 光滑部分Ⅰ(由于迭代能起光滑误差的作用)。已知初值$u_M^{(n)}$，在 Ω_M上作 ν_1次光滑，得

$\overline{u_M^{(n)}} := \text{Relax}^{\nu_1}(u_M^{(n)}, L_M, f_M)$

2° 粗网修正

（1）计算亏损量

$d_M^{(n)} := f_M - L_M \overline{u}_M^{(n)}$

（2）限制亏损量

$d_{M-1}^{(n)} := I_M^{M-1} d_M^{(n)}$

（3）在 Ω_{M-1}上近似计算亏损方程

$L_{M-1} v_{M-1}^{(n)} = d_{M-1}^{(n)}$

求修正量$v_{M-1}^{(n)}$的近似值$\widehat{v}_{M-1}^{(n)}$。计算时以零网格函数作为初始近似值，作 M-网格法的 $r(\geqslant 1)$型迭代（见图7.18.34）。M-网格法用网格 Ω_{M-1}, Ω_{M-2}, $\cdots$, Ω_0以及相应的网格算子，在最粗的网格上精确求解

$L_0 v_0 = d_0$

（4）插值

$v_M^{(n)} := I_{M-1}^M v_{M-1}^{(n)}$

（5）修正 Ω_M上的节点函数值

$\widehat{u}_M^{(n)} = \overline{u}_M^{(n)} + v_M^{(n)}$

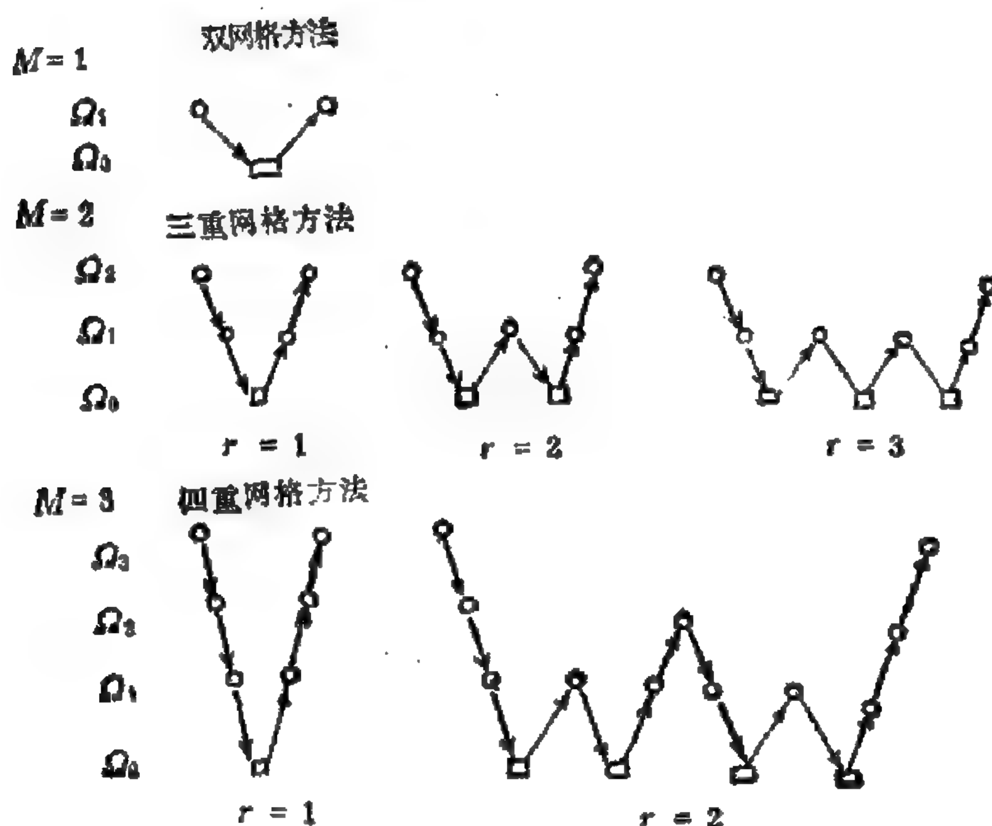
3° 光滑部分Ⅱ．在 Ω_M上作 $\nu_2(\geqslant 0)$次光滑得

$u_M^{(n+1)} := \text{Relax}^{\nu_2}(\widehat{u}_M^{(n)}, L_M, f_m)$

粗网修正中的 r 是循环参数，一般 $r=1, 2$ 或 3。当 $r>3$ 时，多重网格方法则不太有效。

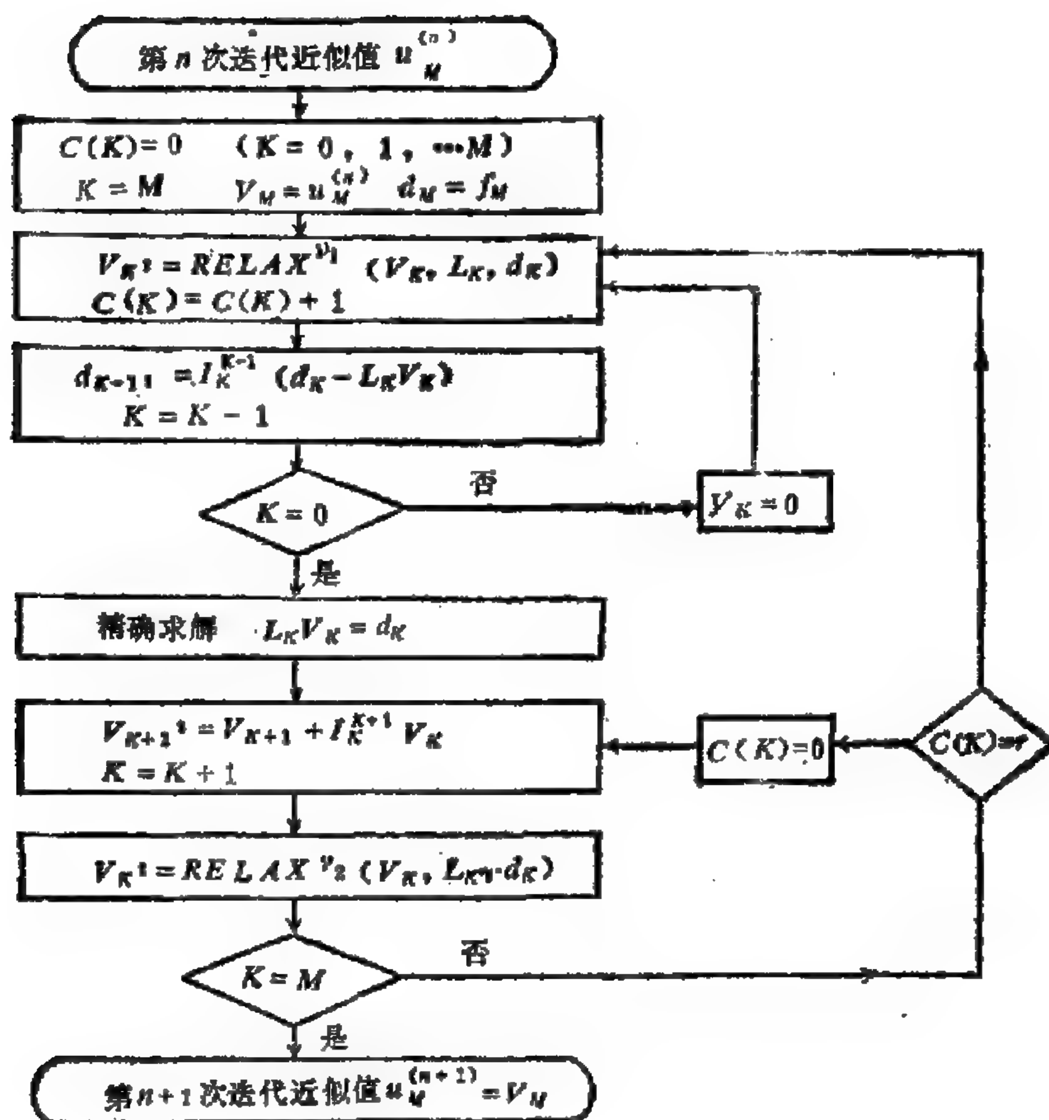
用图（7.18.19）中的符号分别表示 $M=1, 2, 3$ 时 $r=1, 2, 3$ 的多重网格方法的一个循环（或称一个迭代步），则不同 M 值和不同 r 值的 MG 循环如图（7.18.34）所示，称 $r=1$ 为 V 循环，称 $r=2$ 为 W 循环。

7.18.34 图



图（7.18.35）是一个 MG 迭代步的框图，其中 $0\leqslant C(k)\leqslant 1$ 是一个开关参数，控制什么时候转粗网以及什么时候返回到细网。

7.18.35 图 求解 $L_M u_M=f_M(M\geqslant 1)$ 的一个 MG 迭代步框图



MG 迭代公式是

7.18.36 $u_M^{(n+1)}=A_M u_M^{(n)}+G_M[f_M]$

其中 $A_1:=S_1^{\nu_2}(I_1-I_0^1 L_0^{-1} I_1^0 L_1)S_1^{\nu_1}$

7.18.37 $A_k:=S_k^{\nu_2}(I_k-I_{k-1}^k(I_{k-1}-A_{k-1}^r)L_{k-1}^{-1}I_k^{k-1}L_k)S_k^{\nu_1}$

$(k=2,3,\cdots,M)$

而 Ω_{M-1}和 Ω_M 的双网格算子为

7.18.38 $A_M^{M-1}:=S_M^{\nu_2}(I_M-I_{M-1}^M L_{M-1}^{-1} I_M^{M-1} L_M)S_M^{\nu_1}$

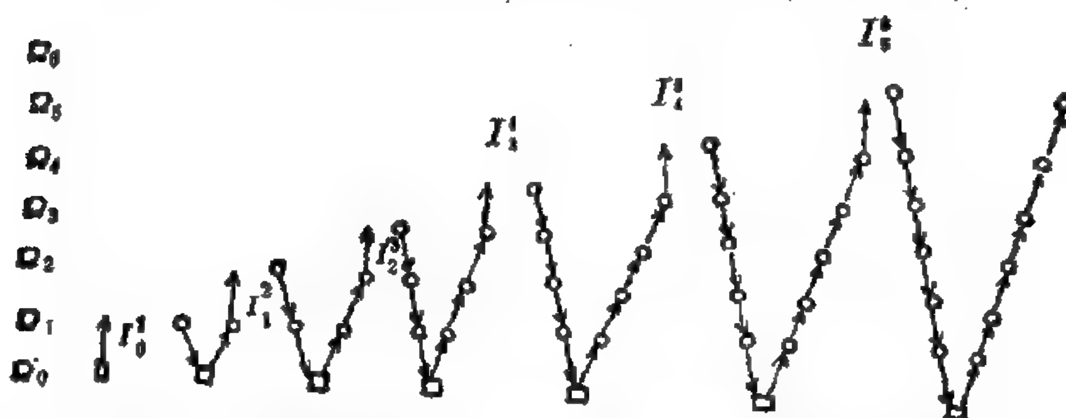
(7.18.37) 和 (7.18.38)之间的差别在于用$(I_{M-1}-A_{M-1}^r)L_{M-1}^{-1}$代替$L_{M-1}^{-1}$，这说明 (7.18.37)是用$\Omega_{M-1}$，$\Omega_{M-2}$，$\cdots\Omega_0$ 这 M 重网格的 r 型MG迭代去近似求解$L_{M-1}v_{M-1}=d_{M-1}$。

在使用线性 MG 方法计算中要证明 MG 方法的收敛性，首先应证明双网格方法的收敛性。一般情况下，若双网格方法收敛得很好，则相应的 W 循环 MG 方法亦收敛得很好。

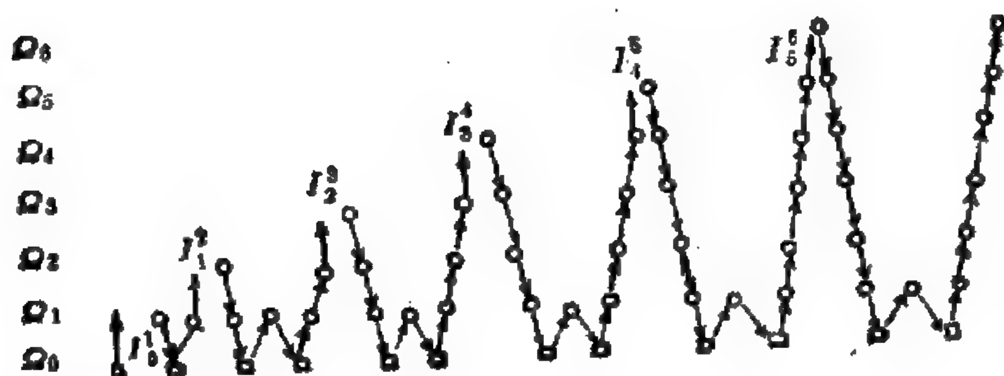
7.18.4 完整的多重网格方法

完整的多重网格方法(Full Multigrid Technique)简称 FMG 方法，它是 MG 方法与**套迭代**（Nested Iteration）相结合的产物。所谓套迭代就是粗网为细网提供良好的近似，这是古典的方法不可能做到的。FMG 方法的计算步骤如图（7.18.39），图（7.18.40）和图（7.18.41）所示。

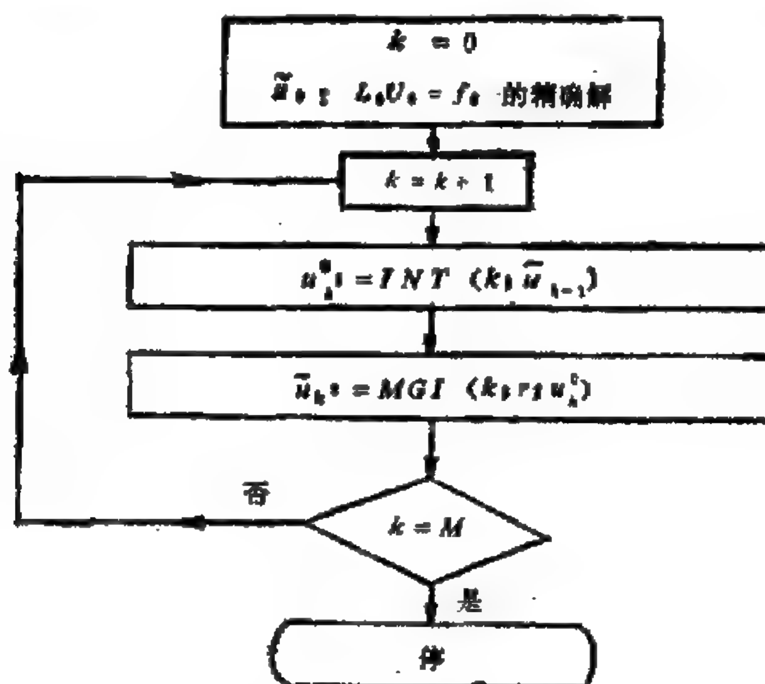
7.18.39 图 V 循环 MG 法与套迭代相结合的 FMG 方法



7.18.40 图 W 循环 MG 法与套迭代相结合的 FMG 方法



7.18.41 图 W 循环MG法与套迭代相结合的FMG方法



在图（7.18.41）中，INT表示插值，MGI表示在$\Omega_k(k=0,1,2,\cdots)$上用r型MG迭代求解方程$L_ku_k=f_k(k=0,1,2\cdots)$；$r=1$为$V$循环，$r=2$为$W$循环。当$M=6,r=1$时，图(7.18.41)即为图(7.18.39)所示的FMG过程；当$M=6$，$r=2$时图（7.18.41）即为图（7.18.40)所示的FMG过程。

7.19

7.19 行处理方法

行处理方法（Row Action Methods）是一类用途很广的迭代方法。由于解奇异线性方程组时它永远收敛，而且计算公式简单，节省存贮空间，因此具有一般直接法和迭代法两者的优点。同时，行处理方法可用于最小二乘问题、广义逆矩阵A^+、线性规划的可行性解、凸规划和非线性方程组的解，已被很多实际应用领域采用。

对于非奇异线性方程组

7.19.1 $Ax=b$

设 $A\in R^{n\times n}$ 是非奇异矩阵，$\alpha_i=(a_{i1},a_{i2},\cdots,a_{in})$为 A 的第i个行向量。方程（7.19.1）可以改写成：

7.19.2 $(\alpha_i^T,x)=b_i \quad (i=1,2,\cdots n)$

其中 b_i 是向量 b 的第 i 个分量，而（·，·）代表点积。为了方便起见，永远假设

7.19.3 $(\alpha_i^T,\alpha_i^T)=1$

（7.19.2）式中每个方程代表 n 维空间中的一个超平面，而 α_i^T 是其法线方向。

假定已知第 k 次迭代值 x_k，为求 x_{k+1}，取一固定的 i 并令

7.19.4 $x_{k+1}-x_k=\alpha\alpha_i^T$

这个式子代表 x_{k+1} 取于从 x_k 出发第 i 个超平面的法线方向。x_{k+1} 应满足

$(\alpha_i^T,x_{k+1})=b_i$

则有

7.19.5 $x_{k+1}=x_k+(b_i-(\alpha_i^T,x_k))\alpha_i^T$

这样，行处理方法的一般计算公式为：

7.19.6 $\begin{cases} x_0 & \text{任取} \\ x_{k+1}=x_k+(b_i-(\alpha_i^T,x_k))\alpha_i^T & (k=0,1,2,\cdots) \\ i=k(\bmod n) \end{cases}$

更一般的行处理方法为

7.19.7 $\begin{cases} x_0 & \text{任取} \\ x_{k+1}=x_k+r(b_i-(\alpha_i^T,x_k))\alpha_i^T & (k=0,1,2,\cdots) \\ \varepsilon_2\leqslant r\leqslant 2-\varepsilon_1 \end{cases}$

其中 $\varepsilon_1>0$，$\varepsilon_2>0$（$\varepsilon_1,\varepsilon_2$ 为小量）。由于 i 的取法有很多种，故产生出不同的行处理方法。（7.19.6）或（7.19.7）一直迭代到 x_k 满足预先给定的精确度为止。

7.20

7.20 压缩存储技巧

对于大型稀疏矩阵，为节省计算机内存，一般只存储它的非零元素，极少量的零元素和一些必要的信息，这就是通常所说的压缩存储法(Packing Storage Scheme)。

7.20.1 按行随机存储非对称稀疏矩阵

对于循环边界条件问题的系数矩阵

$$A=\begin{pmatrix} a_{11} & a_{12} & & & & a_{1,100}\\ a_{21} & a_{22} & a_{23} & & & \\ & a_{32} & a_{33} & a_{34} & & \\ & & \ddots & \ddots & \ddots & \\ & & & & & a_{99,100}\\ a_{100,1} & & & & a_{100,99} & a_{100,100}\end{pmatrix} \tag{7.20.1}$$

用三个一维数组来存储 A 的非零元素及相应的信息。

AA(300)　存储 A 的 300 个非零元素。

JA(300)　300 个非零元素在矩阵 A 中相应的列号。

ISTART(101)　每行第一个非零元素在数组 A 中的位置。最后一个元素是 AA 和 JA 的数组界 +1。

AA $=(a_{11},a_{12},a_{1,100},a_{21},a_{22},a_{23},a_{32},a_{33},a_{34},\cdots a_{100,1},$
$a_{100,99},a_{100,100})$

JA$=$(1,2,100,1,2,3,2,3,4,…,1,99,100,)

ISTART$=$(1,4,7,…,298,301)

例如，找对角线元素a_{22}、$a_{22}\neq 0$时在第2行第1个非零元素和第3行第1个非零元素位置之间，即在ISTART(2)$=$4

和 ISTART(3)=7 之间，一定能找到 a_{22} 的列指标 JA(5)=2，所以 a_{22} 在 AA(5)中。

7.20.2 例 按行随机存储下列非对称矩阵

1°

$$A=\begin{pmatrix} 5 & 0 & -2 & 0 & 0 \\ 0 & 7 & 0 & 1 & 0 \\ 3 & 0 & 2 & 0 & 1 \\ 0 & -5 & 6 & 0 & 0 \\ 0 & 0 & 1 & 0 & 4 \end{pmatrix}$$

AA=(5,−2,7,1,3,2,1,−5,6,1,4)

JA=(1,3,2,4,1,3,5,2,3,3,5)

ISTART=(1,3,5,8,10,12)

2°

$$A=\begin{pmatrix} -2.4 & 7.6 & 0 & 0 & 0 & 8.9 \\ 3.8 & -0.4 & 0 & 2.9 & 0 & 0 \\ 0 & 0 & 3.1 & -5.6 & 11.3 & 0 \\ -1.4 & 0 & 0 & 6.7 & 0 & 4.9 \\ 3.8 & 1.4 & 0 & 0 & -2.9 & 0 \\ 7.2 & 0 & 0 & 0 & -1.6 & 5.3 \end{pmatrix}$$

AA=(−2.4, 7.6, 8.9, 3.8, −0.4, 2.9, 3.1, −5.6, 11.3, −1.4, 6.7, 4.9, 3.8, 1.4, −2.9, 7.2, −1.6, 5.3)

JA=(1, 2, 6, 1, 2, 4, 3, 4, 5, 1, 4, 6, 1, 2, 5, 1, 5, 6)

ISTART=(1,4,7,10,13,16,19)

7.20.2 按行压缩存储对称矩阵

用二个数组 *AA* 和 *JPD* 按行存储 *n* 阶对称矩阵的下三角元素，以五阶矩阵为例。

设

$$A=\begin{pmatrix} a_{11} & & & & \\ a_{21} & a_{22} & & & \\ & a_{32} & a_{33} & & \\ & a_{42} & 0 & a_{44} & \\ & a_{52} & a_{53} & 0 & a_{55} \end{pmatrix}$$

$AA=(a_{11}, a_{21}, a_{22}, a_{32}, a_{33}, a_{42}, 0, a_{44}, a_{52}, a_{53}, 0, a_{55})$

JPD=(1, 3, 5, 8, 12)

数组 AA 中存放矩阵 A，每一行从最左边的非零元素到对角线之间的所有元素，包括对角线元素及夹在中间的零元素。JPD 数组存放矩阵A的对角线元素在数组AA 中的位置。

如果找 a_{53}，JPD(5)－(5－3)＝12－2＝10>JPD(4)＝8，所以 a_{53} 存放在 $AA(10)$中. 如果找 a_{ij}，若

$JPD(i)-(i-j)>JPD(i-1)$

则 $JPD(i)-(i-j)$是 a_{ij} 在 AA 中的位置，这说明 a_{ij} 在第 i 行的第 1 个非零元素的右边. 否则说明在左边，但是左边不可能有第 i 行的非零元素，故 $a_{ij}=0$.

7.20.3 例 压缩存储下列对称矩阵

1° A 是 100×100 的对称阵

$$A=\begin{pmatrix} 2 & -1 & & & & \\ -1 & 2 & -1 & & & \\ & -1 & 2 & -1 & & \\ & & \cdots & & & \\ & & & -1 & 2 & -1 \\ & & & & -1 & 2 \end{pmatrix}$$

AA＝(2,－1,2,－1,2,－1,…,2,－1,2),共199个元素.

JPD＝(1,3,5,7,9,…,197,199),共100个元素.

2° 设

$$A=\begin{pmatrix} 5 & 3 & 0 & 1 & -6 \\ 3 & 7 & 0 & 2 & 0 \\ 0 & 0 & 2 & 0 & 4 \\ 1 & 2 & 0 & 6 & -1 \\ -6 & 0 & 4 & -1 & 11 \end{pmatrix}$$

则

AA=(5,3,7,2,1,2,0,6,-6,0,4,-1,11)

JPD=(1,3,4,8,13)

7.20.3 带型对称稀疏矩阵的存储

用一个长方形二维数组存储矩阵 A 下三角部分中含非零元素的斜线。设

$$A=\begin{pmatrix} a_{11} & a_{12} & & a_{14} & & & \\ a_{21} & a_{22} & & & & & \\ & & a_{33} & & a_{35} & a_{36} & \\ a_{41} & & & a_{44} & a_{45} & & \\ & & a_{53} & a_{54} & a_{55} & & a_{57} \\ & & a_{63} & & & a_{66} & a_{67} \\ & & & & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

用二维数组 AA(7，4)存储带型(Banded)矩阵 A。

$$AA=\begin{pmatrix} & & & a_{11} \\ & & a_{21} & a_{22} \\ & & & a_{33} \\ a_{41} & & & a_{44} \\ & a_{53} & a_{54} & a_{55} \\ a_{63} & & & a_{66} \\ & a_{75} & a_{76} & a_{77} \end{pmatrix}$$

或者用四个一维数组存储，又简单又节省计算机时间，如

$A_1=(0,0,0,a_{41},0,a_{63},0)$

$A_2=(0,0,0,0,a_{53},0,a_{75})$

$A_3=(0,a_{21},0,0,a_{54},0,a_{76})$

$A_4=(a_{11},a_{22},a_{33},a_{44},a_{55},a_{66})$

8　第8章　矩阵的特征值和特征向量

8.1　8.1　引　言

对于标准(Standard)特征值问题

8.1.1　$Ax=\lambda x$

其中 $A\in R^{n\times n}$，λ 是矩阵 A 的特征值，x 是相应的特征向量，选用什么样的方法去求解，取决于矩阵 A 的类型以及是求部分特征值还是全部特征值的要求。实践中遇到的特征值问题是多种多样的，要求各不相同，因此应根据具体情况选择有效的方法。

1°　如果求实矩阵 A 的按模最大的特征值和相应的特征向量，可选用**幂法**(Power Method)。

2°　如果求实矩阵 A 的按模最小的特征值和特征向量，可选用**反幂法**(Inverse Power Method)。

3°　当 A 是大型稀疏矩阵(例如上千阶的绝大多数元素为零的矩阵)或带型矩阵，如果求其按模最大的若干个特征值及相应的特征向量，可选用**同时迭代法**(Simultaneous Iteration)(**子空间迭代法**)或以相似变换为基础的**压缩方法**(Packing Method)。如果求其按模最小的若干个特征值及相应的特征向量，则可选用**反同时迭代法**(Inverse Simultaneous Iteration)。

4°　当 A是非对称实矩阵时，求其全部特征值和特征向量，可先用 Givens(吉文斯)方法或 Householder(豪斯荷尔德)方法把 A 化为上Hessenberg(赫申伯格)矩阵，然后用QR法求特征值，再用反幂法求出相应的特征向量。

5° 如果A是实对称矩阵,求其全部特征值和特征向量,有两种途径.第一种途径是先用 Givens 方法或Householder方法把实对称矩阵化为对称的三对角矩阵，然后选用 **QR 法**或 **QL 法**或 Sturm **序列和二分法**求对称三对角阵的特征值，再用反幂法求相应的特征向量。第二种途径是选用 Jacobi(雅可比)方法，直接用初等相似变换把实对称矩阵变成对角矩阵，对角线元素即 A 的特征值。

对于**广义**(Generalized)特征值问题

8.1.2 $A\boldsymbol{x}=\lambda B\boldsymbol{x}$ 或

8.1.3 $AB\boldsymbol{x}=\lambda\boldsymbol{x}$

其中 A 为对称阵，B 为正定阵。求解时，先用Cholesky(乔莱斯基) 方法分解 B，然后化为对称矩阵的特征值问题，选取行之有效的方法。

8.2

8.2 幂 法

在许多实际应用中，通常不需要求出矩阵的全部特征值和特征向量，而只需要求出按模最大的特征值.

设 A 是 $n\times n$ 矩阵，它的特征值满足

$$|\lambda_1|>|\lambda_2|\geqslant|\lambda_3|\geqslant\cdots\geqslant|\lambda_n|$$

其中 λ_1 为按模最大的特征值。假定存在 n 个线性无关的特征向量 $\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots\boldsymbol{x}_n$，因而任何一个非零向量 $\boldsymbol{u}_0$ 都能表示成它们的线性组合

$$\boldsymbol{u}_0=\sum_{i=1}^{n}\alpha_i\boldsymbol{x}_i$$

其中 α_i 是不全为零的数。

定义迭代格式

$$\boldsymbol{u}_k=A\boldsymbol{u}_{k-1}\quad(k=1,2,\cdots)$$

其中 $\boldsymbol{u}_0$ 为任意非零初始向量。于是，对于充分大的 k,

$$
\begin{aligned}
\boldsymbol{u}_k &= A\boldsymbol{u}_{k-1} = A^2\boldsymbol{u}_{k-2} = \cdots = A^k\boldsymbol{u}_0 \\
&= \sum_{i=1}^{n} \alpha_i \lambda_i{}^k \boldsymbol{x}_i \\
&= \lambda_1{}^k \left\{ \alpha_1 \boldsymbol{x}_1 + \sum_{i=2}^{n} \alpha_i (\lambda_i/\lambda_1)^k \boldsymbol{x}_i \right\} \\
&= \lambda_1{}^k \{\alpha_1 \boldsymbol{x}_1 + \varepsilon_k\}
\end{aligned}
$$

其中 ε_k 是一个分量很小的向量。因而，向量 $\boldsymbol{u}_k$ 是对非规范化特征向量 $\boldsymbol{x}_1$ 的一个近似。这是迭代计算按模最大特征值的简单幂法(Power Method)的基础。

对于任一 i，当 $k\to\infty$ 时，有

$$
\frac{(\boldsymbol{u}_{k+1})_i}{(\boldsymbol{u}_k)_i} = \lambda_1 \left\{ \frac{\alpha_1(\boldsymbol{x}_1)_i + (\varepsilon_{k+1})_i}{\alpha_1(\boldsymbol{x}_1)_i + (\varepsilon_k)_i} \right\} \longrightarrow \lambda_1
$$

8.2.1 幂法的迭代公式

从任意初始向量 $\boldsymbol{v}_0$ 开始，$\boldsymbol{u}_0 = \boldsymbol{v}_0$，按下列公式迭代

$$
\boldsymbol{v}_1 = A\boldsymbol{u}_0 \quad , \quad \boldsymbol{u}_1 = \frac{\boldsymbol{v}_1}{\max\limits_j |v_{1j}|} = \frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_\infty}
$$

$$
\boldsymbol{v}_2 = A\boldsymbol{u}_1 \quad , \quad \boldsymbol{u}_2 = \frac{\boldsymbol{v}_2}{\|\boldsymbol{v}_2\|_\infty}
$$

$$
\cdots
$$

$$
\boldsymbol{v}_k = A\boldsymbol{u}_{k-1} \quad , \quad \boldsymbol{u}_k = \frac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|_\infty}
$$

$$
\cdots
$$

重复迭代至 $\max\limits_j |v_{kj} - v_{k-1,j}| < \varepsilon$ 止，ε 为预先给定的误差量。有

$$
\textbf{8.2.2} \qquad \|\boldsymbol{v}_k\|_\infty \approx \lambda_1 \quad , \quad \boldsymbol{u}_k \approx \frac{\boldsymbol{x}_1}{\|\boldsymbol{x}_1\|_\infty}
$$

λ_1 是矩阵 A 的按模最大的特征值，$\boldsymbol{x}_1$ 为相应的特征向量，

$\frac{x_1}{\|x_1\|_\infty}$为规范化的特征向量，$v_k$，$u_k$ 为迭代向量，用 j 表示它们的分量下标。迭代向量规范化的目的是防止 v_k 的各个非零分量随着迭代的进行趋于无穷（或趋于零），从而造成"溢出"。

收敛速度取决于比值$\left|\frac{\lambda_2}{\lambda_1}\right|$，比值愈小收敛愈快，当$\left|\frac{\lambda_2}{\lambda_1}\right|$接近1时，收敛很可能非常地慢。

8.2.3 例 求矩阵

$$A=\begin{pmatrix} 2 & 3 & 2 \\ 10 & 3 & 4 \\ 3 & 6 & 1 \end{pmatrix}$$

的按模最大的特征值，要求 $\varepsilon=0.0005$。

解 取 $v_0=(0,0,1)^T$，有

K	u_k^T			$\|v_k\|_\infty$
	u_{k_1}	u_{k_2}	u_{k_3}	
0	0	0	1	1
1	0.5	1.0	0.25	4.0
2	0.5	1.0	0.8611	9.0
3	0.5	1.0	0.7306	11.44
4	0.5	1.0	0.7535	10.9224
5	0.5	1.0	0.7493	11.0140
6	0.5	1.0	0.7501	10.9972
7	0.5	1.0	0.7500	11.0004
8	0.5	1.0	0.7500	11.0000

因为矩阵 A 的三个精确特征值是 $\lambda_1=11$，$\lambda_2=-3$，

$\lambda_3=-2$，比值 $\left|\frac{\lambda_2}{\lambda_1}\right|\approx 0.27$ 较小，所以迭代 8 次可得到主特征值的精确值 $|\lambda_1|=11.000$，收敛速度比较快。

8.3

8.3 幂法的加速方法

8.3.1 原点平移法

幂法的收敛速度取决于矩阵 A 的次大特征值与最大特征值之比值 $r=\left|\frac{\lambda_2}{\lambda_1}\right|$. r 愈小收敛得愈快，为此，引进矩阵

$$B=A-pI$$

其中 p 为可选择参数. 显然，若 A 的特征值为 λ_1, λ_2, $\cdots$, λ_n，则 B 的相应特征值为 λ_1-p, λ_2-p, $\cdots$, λ_n-p，而且 A 和 B 有相同的特征向量。

适当选择参数 p，使 λ_1-p 是 B 的按模最大特征值，即

$$|\lambda_1-p|>|\lambda_k-p| \quad (k=2,3,\cdots,n)$$

且使

8.3.1 $$\max_{2\leqslant k\leqslant n}\left|\frac{\lambda_k-p}{\lambda_1-p}\right|<\left|\frac{\lambda_2}{\lambda_1}\right|$$

于是可先用幂法计算 B 的按模最大特征值 $\lambda_1(B)$，其收敛速度比用幂法计算 A 的 λ_1 要快。由于 $\lambda_1(B)=\lambda_1-p$，即求得 $\lambda_1=\lambda_1(B)+p$，故称这种方法为**原点平移法**(Shift of Origin). 对于特征值的某种分布，它是十分有效的. 例如，对于具有特征值 $\lambda_j=15-j(j=1,2,3,4)$ 的四阶矩阵，通常的幂法其收敛速度取决于 $\left|\frac{\lambda_2}{\lambda_1}\right|=0.9285$，显然收敛很慢. 但若作原点平移，选择参数 $p=12$，则对于矩阵 $A-pI$ 其收敛速度取决

于 $\left|\frac{\lambda_2-p}{\lambda_1-p}\right|=0.5$，收敛速度得到提高.

虽然常常能够选择有利的 p 值，使幂法得到加速，但设计一个自动选择适当参数 p 的过程是困难的.

8.3.2 例 用幂法及原点平移法计算矩阵

$$A=\begin{pmatrix}1.0 & 1.0 & 0.5\\1.0 & 1.0 & 0.25\\0.5 & 0.25 & 2.0\end{pmatrix}$$

的最大特征值，取 $p=0.75$.

解 用幂法计算 A 的最大特征值，其计算过程如表(8.3.3).

8.3.3 表

K	u_k^T（规范化向量）	$\|v_k\|_\infty$
0	(1 1 1)	
5	(0.7651 0.6674 1)	2.5587918
10	(0.7494 0.6508 1)	2.5380029
15	(0.7483 0.6497 1)	2.5366259
20	(0.7482 0.6497 1)	2.5365323

然后，采用原点平移法进行加速，因为 $p=0.75$，则有

$$A-pI=\begin{pmatrix}0.25 & 1 & 0.5\\1 & 0.25 & 0.25\\0.5 & 0.25 & 1.25\end{pmatrix}$$

对 $A-pI$ 应用幂法，计算结果如表(8.3.4).

8.3.4 表

K	α_k^T（规范化向量）			$\|v_k\|_\infty$
0	1	1	1	
5	0.7516	0.6522	1	1.7914011
8	0.7484	0.6499	1	1.7869162
9	0.7483	0.6497	1	1.7866587
10	0.7482	0.6497	1	1.7865914

由此得到 $A-pI$ 的最大特征值为 $\mu_1\approx 1.7865914$，A 的最大特征值 $\widetilde{\lambda_1}=\mu_1+0.75=2.5365914$。

A 的最大特征值和特征向量的真值是 $\lambda_1=2.5365258$，$x_1=(0.74822116, 0.64966116, 1)^T$。从表(8.3.3)(8.3.4)与真值对照比较可看出，加速后迭代 10 次得到的结果比表(8.3.3)中迭代 15 次的结果还好。

8.3.2 Aitken 加速法

由幂法的迭代公式有

$\mu_k=\|v_k\|_\infty$

当 k 充分大时逼近于矩阵 A 的最大特征值 λ_1。由于序列 $\{\mu_k\}$ 是线性收敛于 λ_1 的，因此，对于充分大的 k，有如下的近似关系式

8.3.5 $$\frac{\mu_{k+2}-\lambda_1}{\mu_{k+1}-\lambda_1}\approx\frac{\mu_{k+1}-\lambda_1}{\mu_k-\lambda_1}$$

从而

8.3.6 $$\lambda_1\approx\frac{\mu_k\mu_{k+2}-\mu_{k+1}^2}{\mu_k-2\mu_{k+1}+\mu_{k+2}}\approx\widetilde{\mu}$$

显然，由已知的三个迭代值 μ_k，μ_{k+1}，μ_{k+2} 就可从(8.3.6)式算得新的特征值估计。同样，可以求得特征向量的新的估

值：

8.3.7 $\widetilde{u}_i=\dfrac{(u_k)_i(u_{k+2})_i-(u_{k+1})_i^2}{(u_k)_i-2(u_{k+1})_i+(u_{k+2})_i}\quad(i=1,2,\cdots,n)$

这种加速方法被称为 Aitken(埃特金)加速法。

8.3.3 Rayleigh 商加速

设 A 为 n 阶实对称矩阵，对于任一非零向量 $\boldsymbol{x}$，称

8.3.8 $R(\boldsymbol{x})=\dfrac{(A\boldsymbol{x},\boldsymbol{x})}{(\boldsymbol{x},\boldsymbol{x})}$

为对应于向量 $\boldsymbol{x}$ 的Rayleigh(瑞利)商。

若 λ_1 是 A 的最大特征值，当且仅当 $\boldsymbol{x}$ 是对应于 λ_1 的特征向量，则

$$\lambda_1=\max\frac{(A\boldsymbol{x},\boldsymbol{x})}{(\boldsymbol{x},\boldsymbol{x})}$$

Rayleigh 商主要为着加速用幂法获得实对称矩阵的最大特征值的收敛性。因为由幂法的迭代序列有

8.3.9 $\dfrac{(\boldsymbol{u}_{k+1})_i}{(\boldsymbol{u}_k)_i}=\lambda_1+O\left(\left(\dfrac{\lambda_2}{\lambda_1}\right)^k\right)$

而 Rayleigh 商

8.3.10 $\dfrac{(A\boldsymbol{u}_k,\boldsymbol{u}_k)}{(\boldsymbol{u}_k,\boldsymbol{u}_k)}=\lambda_1+O\left(\left(\dfrac{\lambda_2}{\lambda_1}\right)^{2k}\right)$

以上两式相比较，显然，Rayleigh 商逼近 λ_1 的速度更快。

8.3.11 Rayleigh 商加速幂法的迭代公式

任取非零初始向量 u_0

$$\begin{cases}\boldsymbol{v}_k=A\boldsymbol{u}_{k-1}\\[2mm]\mu_k=\dfrac{(A\boldsymbol{u}_{k-1},\boldsymbol{u}_{k-1})}{(\boldsymbol{u}_{k-1},\boldsymbol{u}_{k-1})}\quad(k=1,2,\cdots)\\[2mm]\boldsymbol{u}_k=\dfrac{\boldsymbol{v}_k}{\mu_k}\end{cases}$$

迭代一直进行到 $|\mu_{k+1}-\mu_k|<\varepsilon$ 为止。ε 为预先给定的容许误差值，这时，$\mu_k\approx\lambda_1$，$\boldsymbol{u}_k\approx c\boldsymbol{x}_1$，其中 c 是某个正常数。

8.4

8.4 压缩方法

假定矩阵 A 的特征值 $|\lambda_1|$，$|\lambda_2|$，…，$|\lambda_m|$ 彼此有相当间隔，用幂法求得按模最大的特征值 λ_1，而要计算按模次大的特征值 $\lambda_2,\lambda_3,\cdots,\lambda_m$。在这种情况下，求特征值的最有效的方法，是采用"压缩"方法。压缩(Packing)方法是以相似变换为基础的，从原始矩阵导出一个新矩阵，它仅含有原始矩阵 A 剩下的未知特征值 $\lambda_2,\lambda_3,\cdots,\lambda_m$。重复应用这个方法，能顺次地把次大特征值及对应的特征向量计算出来。

假定已经算出矩阵 A 的 λ_1 和对应的特征向量 $\boldsymbol{x}_1$，然后构造一个非奇异矩阵 R_1，便得

8.4.1 $R_1\boldsymbol{x}_1=k\boldsymbol{e}_1$

其中 $k\neq0$。令 $A_1=A$，则

$R_1A_1(R_1^{-1}R_1)\boldsymbol{x}_1=\lambda_1R_1\boldsymbol{x}_1$

于是

8.4.2 $R_1A_1R_1^{-1}\boldsymbol{e}_1=\lambda_1\boldsymbol{e}_1$

因而可写成

8.4.3 $A_2=R_1A_1R_1^{-1}=\begin{pmatrix}\lambda_1 & \boldsymbol{r}^T\\ 0 & B_2\end{pmatrix}$

其中 B_2 是 $n-1$ 阶矩阵，$\boldsymbol{r}$ 为有 $n-1$ 个分量的向量。由于相似变换，A_2 与 A_1 有同样的特征值，所以 B_2 具有特征值 $\lambda_2,\lambda_3,\cdots,\lambda_n$。从而可以计算 B_2 的最大特征值也是 A 的次大特征值 λ_2，以及满足

8.4.4 $B_2\boldsymbol{y}_2=\lambda_2\boldsymbol{y}_2$

的特征向量 $\boldsymbol{y}_2$。

为求 A 的与 λ_2 对应的特征向量 $\boldsymbol{x}_2$，设 $\boldsymbol{z}_2$ 是 A_2 对应

于 λ_2 的特征向量，那么

8.4.5 $\begin{bmatrix} \lambda_1 & \boldsymbol{r}^T \\ 0 & B_2 \end{bmatrix} \boldsymbol{z}_2 = \lambda_2 \boldsymbol{z}_2$

利用（8.4.4）式，可以取

8.4.6 $\boldsymbol{z}_2 = \begin{bmatrix} a \\ \boldsymbol{y}_2 \end{bmatrix}$

其中 a 是一个数量，由

8.4.7 $(\lambda_1 - \lambda_2)a + \boldsymbol{r}^T \boldsymbol{y}_2 = 0$

给定，最后得到

8.4.8 $\boldsymbol{x}_2 = R_1^{-1} \boldsymbol{z}_2$

继续用这种方法，就可得到 A 的其余次大特征值和对应的特征向量。

构造 R_1 的最简单的办法是令

8.4.9 $R_1 = L_1 I_{1,p}$

其中 L_1 是初等下三角阵，$I_{1,p}$ 是一个置换阵，p 是使得按模来说 $(\boldsymbol{x}_1)_p$ 是 $\boldsymbol{x}_1$ 的最大元素，R_1 使得

$L_1 I_{1,p} \boldsymbol{x}_1 = k\boldsymbol{e}_1$

令 $\boldsymbol{y} = I_{1,p} \boldsymbol{x}_1$，则有

$L_1 \boldsymbol{y} = k\boldsymbol{e}_1$

其中

$$L_1 = \begin{bmatrix} 1 & & & \\ -y_2/y_1 & 1 & & \\ \vdots & & \ddots & \\ -y_n/y_1 & & & 1 \end{bmatrix}, \quad I_{1,p} = \begin{bmatrix} 0 & & & 1 & & & \\ & 1 & & & & & \\ & & \ddots & & & & \\ & & & 1 & & & \\ 1 & & & & 0 & & \\ & & & & & 1 & \\ & & & & & & \ddots \\ & & & & & & & 1 \end{bmatrix} p$$

$\boldsymbol{y} = (y_1, y_2, \cdots, y_n)^T = (x_{1p}, x_{12}, \cdots, x_{11}, \cdots, x_{1n})^T$

$k = y_1 = (\boldsymbol{x}_1)_p$

由于引入置换矩阵 $I_{1,p}$，使 R_1 的元素按模不超过 **1**，所以保证压缩算法是稳定的。

8.4.10 **例** 矩阵

$$A_1=\begin{pmatrix} 2 & 3 & 2 \\ 10 & 3 & 4 \\ 3 & 6 & 1 \end{pmatrix}$$

的最大特征值 $\lambda_1=11.0$，对应的特征向量 $x_1=(0.5, 1.0, 0.75)^T$，试用压缩法求 A_1 的其余特征值。

解 x_1 中第 2 个分量按模最大，因此，$p=2$。交换矩阵是 I_{12}，于是有

$$y=I_{12}x_1=(1.0, 0.5, 0.75)^T$$

$$=\begin{pmatrix} 1 & & \\ -0.5 & 1 & \\ -0.75 & 0 & 1 \end{pmatrix}$$

$$A_2=L_1I_{1,2}A_1I_{1,2}L_1^{-1}$$

$$=\begin{pmatrix} 1 & & \\ -0.5 & 1 & \\ -0.75 & 0 & 1 \end{pmatrix}\begin{pmatrix} 3 & 10 & 4 \\ 3 & 2 & 2 \\ 6 & 3 & 1 \end{pmatrix}\begin{pmatrix} 1 & & \\ 0.5 & 1 & \\ 0.75 & 0 & 1 \end{pmatrix}$$

$$=\begin{pmatrix} 11 & 10 & 4 \\ 0 & -3 & 0 \\ 0 & -4.5 & -2 \end{pmatrix}$$

由(8.4.3)

$$B_2=\begin{pmatrix} -3 & 0 \\ -4.5 & -2 \end{pmatrix}$$

由此可知，其余的特征值为 -3 和 -2。

8.5

8.5 实对称矩阵的同时迭代法（子空间迭代法）与反同时迭代法

实对称矩阵的**同时迭代法**(Simultaneous Iteration)基于若干个**试验向量**(Trial Vectors)进行同时迭代，迭代收敛后，同时得到实对称矩阵 A 的若干个特征值和相应的特征向量.

8.5.1 基本原理

设 A 是一个 n 阶实对称矩阵，其特征值满足

$|\lambda_1| \geqslant |\lambda_2| \geqslant \cdots \geqslant |\lambda_n|$

而对应的特征向量满足

8.5.1 $\quad x_i^T x_j = \delta_{ij} = \begin{cases} 1, & i=j \\ 0, & i \neq j \end{cases}$

设 $n \times m$ 矩阵

8.5.2 $\quad U = [u_1, u_2, \cdots, u_m]$

其中 $u_i (i=1,2,\cdots m)$ 是 m 个试验列向量，$1 < m \leqslant n$，它们分别是矩阵 A 特征向量 $x_1, x_2, \cdots, x_m$ 的近似向量而且是规范化的，因此有

8.5.3 $\quad U^T U = I_m$

用

8.5.4 $\quad V = AU$

定义 $n \times m$ 矩阵

$V = [v_1, v_2 \cdots, v_m]$

如果试验向量 $u_i (i=1,2,\cdots,m)$ 是 A 的精确的特征向量，那么矩阵

8.5.5 $\quad B = U^T V = U^T A U$

是以 A 的 m 个特征值作为对角元素的对角阵. 非对角元素的大小一般取决于向量 u_i 和 v_j 之间的相互作用的程度，即

$u_i^T v_j$ $(i \neq j)$ 接近于零的程度，由于这个原因，称 B 为相互作用矩阵(Interaction Matrix)。

令 $X=[x_1, x_2, \cdots, x_n]$，其中 x_i 是相应于矩阵 A 的特征值 λ_i 的特征向量，则有

8.5.6　$U=XC$

其中 C 是 $n \times m$ 的系数矩阵。由于 $X^TX=I_n$ 及 $U^TU=I_m$，因此，必定有

8.5.7　$C^TC=I_m$

于是，由 $X^TAX=\Lambda$ 有

8.5.8　$B=U^TAU=(XC)^TA(XC)=C^TX^TAXC=C^T\Lambda C$

其中 $\Lambda=\mathrm{diag}(\lambda_1, \lambda_2, \cdots, \lambda_n)$

矩阵 C 不是方阵，可以把（8.5.6）式改写成

8.5.9　$U=X_aC_a+X_bC_b$

其中

$X_a=[x_1, x_2, \cdots, x_m]$

$X_b=[x_{m+1}, x_{m+2}, \cdots, x_n]$

$$C_a=\begin{pmatrix} C_{11} & C_{12} & \cdots & C_{1m} \\ C_{21} & C_{22} & \cdots & C_{2m} \\ \cdots & & & \\ C_{m1} & C_{m2} & \cdots & C_{mm} \end{pmatrix}, \quad C_b=\begin{pmatrix} C_{m+1,1} & \cdots & C_{m+1,m} \\ C_{m+2,1} & \cdots & C_{m+2,m} \\ \cdots & & \\ C_{n1} & \cdots & C_{n,m} \end{pmatrix}$$

C_a 和 C_b 是系数矩阵，C_a 是方阵。于是，由（8.5.8）式，有

8.5.10　$B=C_a{}^T\Lambda_aC_a+C_b{}^T\Lambda_bC_b$

其中

$\Lambda_a=\mathrm{diag}(\lambda_1, \lambda_2, \cdots \lambda_m)$，　$\Lambda_b=\mathrm{diag}(\lambda_{m+1}, \lambda_{m+2}, \cdots \lambda_n)$

这样，方程（8.5.7）就成为

8.5.11　$C_a{}^TC_a+C_b{}^TC_b=I_m$

若假定 C_b 与 C_a 相比是很小的，(8.5.11)式和(8.5.10)

式可近似地写成

8.5.12 $C_a^{-1} \approx C_a^T$

8.5.13 $BC_a^T \approx C_a^T \Lambda_a$

这时可以把 C_a^T 近似地看作 B 的特征向量阵 P，把 B 的特征值近似地看作 A 的 m 个最大特征值．这个结论给出了同时迭代法的基础。

假定把 V 表示成

8.5.14 $V = X_a \Lambda_a C_a + X_b \Lambda_b C_b$

的形式，则在忽略式中的 C_b 并利用（8.5.12）式后，有

$VP \approx VC_a^T \approx X_a \Lambda_a$

因此，

$W = VP$

更接近于 A 的前 m 个特征向量．由于 W 一般不是正交矩阵，所以必须把 W 正交化后构成新的试验向量 U。

8.5.2 同时迭代法的迭代过程

8.5.15 同时迭代法的主要计算步骤如下：

1° 取初始 $U^{(0)} = [\boldsymbol{u}_1^{(0)}, \boldsymbol{u}_2^{(0)}, \cdots \boldsymbol{u}_m^{(0)}]$；

2° 计算 $V = AU^{(k-1)}$；

3° 计算 $B = (U^{(k-1)})^T V$；

4° 计算 B 的特征值和特征向量所构成的矩阵 P；

5° 计算 $W = VP$；

6° 把 W 正规化后构成 $U^{(k)}$；

7° 用 $U^{(k)}$ 和 $U^{(k-1)}$ 相应列向量之间差的范数来判断迭代的收敛性．如果没有满足精度要求，可转到2°，重复此过程直到迭代收敛为止．

步骤 2° 和幂法一样，能增强对应于 $\lambda_1, \lambda_2, \cdots, \lambda_m$ 的特征向量的优势，因此，每次迭代，系数矩阵 C_b 都将被简化．于是，B 的特征值趋向于 A 的 m 个最大特征值，而矩阵 U

的各列收敛于 A 的对应的特征向量.

关于步骤4°，如果 $m \ll n$，计算 B 的特征值比计算 A 的特征值要简单得多，而且任何数值方法都可以使用. 在迭代的后面几步，B 接近于对角阵，可以用(8.7)中的 Jacobi（雅可比）方法对角化.

8.5.16 例 用同时迭代法求矩阵

$$A=\begin{pmatrix}4 & 1 & 4\\ 1 & 10 & 1\\ 4 & 1 & 10\end{pmatrix}$$

的特征值 λ_1，λ_2 和对应的特征向量.

解

1° 取初始 $U^{(0)}=\begin{pmatrix}1 & 0\\ 0 & 1\\ 0 & 0\end{pmatrix}$；

2° 计算 $V=AU^{(0)}=\begin{pmatrix}4 & 1\\ 1 & 10\\ 4 & 1\end{pmatrix}$；

3° 计算 $B=(U^{(0)})^TV=\begin{pmatrix}4.0 & 1.0\\ 1.0 & 10.0\end{pmatrix}$；

4° 计算 B 的特征值和相应的特征向量矩阵 P，B 的特征值 $\mu_1=10.16$，$\mu_2=3.84$，

$$P=\begin{pmatrix}0.16 & 0.99\\ 0.99 & -0.16\end{pmatrix}$$

其中 $P^TP\doteq I$；

5° 计算 $W=VP=\begin{pmatrix}1.63 & 3.79\\ 10.03 & -6.15\\ 1.63 & 3.79\end{pmatrix}=(w_1,w_2)$；

6° 把 W 正规化，即 $\left(\frac{w_1}{\|w_1\|_2}, \frac{w_2}{\|w_2\|_2}\right)$，用 $U^{(1)}$ 表示为

$$U^{(1)}=\begin{bmatrix} 0.16 & 0.69 \\ 0.98 & -0.22 \\ 0.16 & 0.69 \end{bmatrix}$$

用 $U^{(1)}$ 代替 $U^{(0)}$，重复过程 2°—6°，计算结果如表(8.5.17)。记号 $\|\cdot\|_2$ 表示 2 范数（定义 7.2.2）。

8.5.17 表

k	$(U^{(k)})^T$			B		μ_1 μ_2
1	1 0	0 1	0 0	4.0 1.0	1.0 10.1	10.16 3.84
2	0.16 0.69	0.98 −0.22	0.16 0.69	10.67 1.49	1.49 10.33	12.00 9.00
3	0.424 −0.251	0.566 0.824	0.707 −0.508	12.600 −0.431	−0.431 9.391	12.657 9.334
4	0.421 −0.163	0.454 0.889	0.785 −0.427	12.677 −0.014	−0.014 9.348	12.677 9.348
5	0.417 −0.155	0.451 0.891	0.789 −0.427	12.677 −0.0005	−0.0005 9.348	12.677 9.348

8.5.3 反同时迭代法

同时迭代法也可用来求按模最小的 m 个特征值。将同时迭代法计算步骤（8.5.15）中的2°换为按反幂法(8.6.1）进行迭代，有如下的相应公式：

8.5.18 $AV=U^{(k-1)}$

求解 V，其它公式均不变。这种方法称为**反同时迭代法**（Inverse Simultaneous Iteration Method）。

8.6

8.6 反幂法

反幂法（Inverse Power Method）或称逆迭代是计算矩阵特征值和特征向量最有效的方法之一，它常常具有较快的收敛速度。

8.6.1 求按模最小的特征值

反幂法是用幂法(8.2) 计算A^{-1}的按模最大的特征值的方法，即计算A的按模最小的特征值的方法。

设A为n阶非奇异矩阵，其特征值为

$|\lambda_1|\geq|\lambda_2|\geq\cdots\geq|\lambda_n|$

相应的特征向量是$\boldsymbol{x}_1,\boldsymbol{x}_2,\cdots,\boldsymbol{x}_n$。于是，$A^{-1}$的特征值为

$$\left|\frac{1}{\lambda_n}\right| \geq \left|\frac{1}{\lambda_{n-1}}\right| \geq\cdots\geq \left|\frac{1}{\lambda_1}\right|$$

相应的A^{-1}的特征向量为 $\boldsymbol{x}_n,\boldsymbol{x}_{n-1},\cdots,\boldsymbol{x}_1$。因此，计算$A$的按模最小的特征值$\lambda_n$的问题，即是计算 A^{-1} 的按模最大的特征值问题。

8.6.1 反幂法迭代步骤：

1° 任取初始非零向量$\boldsymbol{u}_0=\boldsymbol{v}_0$；

2° 对$\boldsymbol{v}_k=A^{-1}\boldsymbol{u}_{k-1}$

可以通过列主元消去法解方程组

$A\boldsymbol{v}_k=\boldsymbol{u}_{k-1}$

得到$\boldsymbol{v}_k$，或者实现对A的三角分解$A=LU$，其等价于求两个三角形方程组

8.6.2 $L\boldsymbol{y}_k=\boldsymbol{u}_{k-1}$

8.6.3 $U\boldsymbol{v}_k=\boldsymbol{y}_k$

其中L为下三角阵，U为上三角阵，从而得到$\boldsymbol{v}_k$；

3° 由$\boldsymbol{u}_k=\dfrac{\boldsymbol{v}_k}{\|\boldsymbol{v}_k\|_\infty}$

求u_k，重复2°-3°迭代直到$\max_j |v_{kj}-v_{k-1,j}|<\varepsilon$为止，$\varepsilon$为预先给定的精确度，有

$$\|v_k\|_\infty \approx \frac{1}{\lambda_n}$$

$$u_k \approx \frac{x_n}{\|x_n\|_\infty}$$

选取初始向量的一种有效的方法是略去(8.6.2)式，用

8.6.4 $Uv_1 = e$

得出第一步迭代，其中$e^T=(1,1,\cdots,1)$.这相当于选$u_0=Le$.

8.6.2 计算给定的近似特征值相应的特征向量

给定一个用其它方法求得的近似特征值，用反幂法往往可以非常有效地求出其相应的特征向量.

设p是A的某一个特征值λ_j的近似值（但不是λ_j的精确值），且满足

$$|\lambda_j-p|<<|\lambda_i-p| \qquad (i\neq j)$$

显然，$\frac{1}{\lambda_j-p}$是$(A-pI_n)^{-1}$的最大特征值，且相应的特征向量仍是A的相应于λ_j的特征向量x_j.于是可按照反幂法的迭代步骤(8.6.1)计算$(A-pI_n)^{-1}$的最大特征值$\frac{1}{\lambda_j-p}$和相应的特征向量x_j，从而得到A的特征值λ_j以及相应的特征向量x_j.只要将(8.6.1)中的A^{-1}用$(A-pI_n)^{-1}$代替，其余过程完全相同。同样，有

$$\|v_k\|_\infty \approx \frac{1}{\lambda_j-p}$$

8.6.5 $\lambda_j \approx p + \frac{1}{\|v_k\|_\infty}$

8.6.6 $u_k \approx \frac{x_j}{\|x_j\|_\infty}$

且收敛速度由比值

$$r = \max_{\lambda_i \neq \lambda_j} \left| \frac{\lambda_j - p}{\lambda_i - p} \right|$$

确定.因此，只要p是λ_j的一个较好的近似，且A的特征值分离得较好，r一般就很小，常常只要迭代1～2次就收敛.特别对于三对角阵和Hesseberg（赫申伯格）矩阵,求它们对应于一个给定的近似特征值所相应的特征向量时，反幂法是最佳方法之一。

8.6.7 例 用反幂法求

$$A = \begin{pmatrix} 2 & 1 & 0 \\ 1 & 3 & 1 \\ 0 & 1 & 4 \end{pmatrix}$$

的对应于近似特征值$\widetilde{\lambda} = 1.2679$(精确特征值为$\lambda_3 = 3 - \sqrt{3}$)的特征向量（用 5 位浮点数计算)。

解 1° 用列主元素消去法实现$A - \widetilde{\lambda} I_n$的三角分解，即

8.6.8 $P(A - \widetilde{\lambda} I_n) = LU$

其中P为排列阵(Permutation Matrix)，它是由选主元素时进行行交换产生的矩阵（如果不选主元，则$P = I_n$)，$\widetilde{\lambda} = 1.2679$.分解结果如下：

$$P = \begin{pmatrix} 0 & 1 & 0 \\ 0 & 0 & 1 \\ 1 & 0 & 0 \end{pmatrix}$$

$$L = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0.7321 & -0.26807 & 1 \end{pmatrix}$$

$$U=\begin{pmatrix}1 & 1.7321 & 1\\ 0 & 1 & 2.7321\\ 0 & 0 & 0.29405\times10^{-3}\end{pmatrix}$$

2° 按(8.6.4)取初始向量

$U\boldsymbol{v}_1=(1,1,1,)^T$

解得

$\boldsymbol{v}_1=(12\ 692,-9\ 290.3,3\ 400.8)^T$,

$\|\boldsymbol{v}_1\|_\infty=12\ 692$

$$\boldsymbol{u}_1=\frac{\boldsymbol{v}_1}{\|\boldsymbol{v}_1\|_\infty}=(1,-0.73198,0.26795)^T$$

3° $\boldsymbol{v}_2=(A-\tilde{\lambda}I_n)^{-1}\boldsymbol{u}_1$

由(8.6.8)得

$LU\boldsymbol{v}_2=P\boldsymbol{u}_1=(-0.73198,0.26795,1)^T$

先解 $L\boldsymbol{y}_2=(-0.73198,0.26795,1)^T$，得

$\boldsymbol{y}_2=(-0.73198,0.26795,1.6077)^T$

再解$U\boldsymbol{v}_2=\boldsymbol{y}_2$，得

$\boldsymbol{v}_2=(20\ 409,-14\ 940,5\ 468.4)^T$, $\|\boldsymbol{v}_2\|_\infty=20\ 409$

$\boldsymbol{u}_2=(1,-0.73203,\ 0.26792)^T$

4° $\lambda=\tilde{\lambda}+\dfrac{1}{\|\boldsymbol{v}_2\|_\infty}=1.267949$

真值$\lambda_3=3-\sqrt{3}\approx1.2679492$

λ_3对应的特征向量的真值是

$\boldsymbol{x}_3=(1,1-\sqrt{3},2-\sqrt{3})^T\approx(1,-0.73205,0.26795)^T$

由此可见，λ是λ_3的相当好的近似，$\boldsymbol{u}_2$是$\boldsymbol{x}_3$的相当好的近似。

8.7

8.7 Jacobi方法

8.7.1 原理和算法

Jacobi（雅可比）方法是利用初等相似变换计算实对称

矩阵的全部特征值及相应特征向量的一种迭代方法．根据正交变换的理论，任何实对称矩阵都可以用正交相似变换把它化为对角形矩阵，其主对角线元素就是原矩阵的特征值．Jacobi方法选取的是一组Givens（吉文斯）平面旋转变换，用这种变换使矩阵对角化的过程一般是一种无限迭代的过程，然而可以用有限次的变换把矩阵的主对角线以外的元素按模变得很小．

设A是$n\times n$矩阵，$P(i,\ j)$是x_i, x_j平面内的Givens平面旋转矩阵(Plane Rotation Matrix)

$$P(i,j)=\begin{array}{c}\begin{matrix} & & & i & & & j & & & \end{matrix}\\ \begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos\theta & & & \sin\theta & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -\sin\theta & & & \cos\theta & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{pmatrix}\end{array}\begin{matrix} \\ \\ \\ i \\ \\ \\ \\ j \\ \\ \\ \\ \end{matrix} \qquad (8.7.1)$$

其中θ是旋转角．$P(i,j)$是正交矩阵，它和单位矩阵只有在$(i,\ i)$，$(i,\ j)$，$(j,\ i)$，$(j,\ j)$四个位置上的元素不同，PA只改变A的第i行与第j行元素；AP^T只改变A的第i列与第j列元素；$A_1=PAP^T$只改变A的第i行，第j行，第i列，第j列元素．如果用$S(A)$表示A的非对角线元素的平方和，$D(A)$表示A的对角线元素平方和，则有

$D(A_1)=D(A)+2a^2_{ij}$

$S(A_1)=S(A)-2a^2_{ij}$

这说明A_1的对角线元素平方和比A的对角线元素平方和增加$2a^2_{ij}$，而A_1的非对角线元素平方和减少$2a^2_{ij}$．A_1仍然是对称矩阵，还可以继续变换，经有限次变换

$A_m=P_mA_{m-1}P^T_m \qquad (m=1,2,\cdots M)$

后，最后变成（近似）对角阵D，即

$P_M \cdots P_2 P_1 A P_1^T P_2^T \cdots P_M^T \approx D$

由于P_i的正交性$P_i^T P_i = I_n$ $(i=1, 2, \cdots M)$，上式可写成$AR_M^T \approx DR_M^T$

其中$R_M^T = P_1^T P_2^T \cdots P_M^T$，$R_M^T$的列向量是 A 的近似特征向量. D的对角元素是A的近似特征值.

8.7.2 Jacobi算法

原始矩阵和每次旋转后的矩阵都存贮在二维数组 $A(n, n)$中，特征向量存贮在$R(n,n)$ 中，n为矩阵A和R的阶. 设$A=(a_{ij})_n$，$R=(r_{ij})_n$

记号 := 表示赋值.

1° 将I_n中的值传送给数组R；

2° 选主元：考虑矩阵A的对称性，在 A 的主对角线上方的元素中寻找按模最大的元素

$$a_{ij} = \max_{l<k} |a_{lk}|$$

3° 计算平面旋转矩阵$P(i, j)$ 中的元素$\sin\theta$和$\cos\theta$：

当$a_{ii}=a_{jj}$时，取$\theta = \mathrm{sign}(a_{ij})\dfrac{\pi}{4}$，在这种情况下，容易计算$\sin\theta$和$\cos\theta$.

当$a_{ii} \neq a_{jj}$时，

$$\tan 2\theta = \frac{2\tan\theta}{1-\tan^2\theta} = \frac{2a_{ij}}{a_{ii}-a_{jj}} = \frac{1}{c}$$

可求得

$$\tan\theta = -c \pm \sqrt{1+c^2}$$

从两个值中取模数较小的一个值

$$\tan\theta = \begin{cases} -c+\sqrt{1+c^2} = (|c|+\sqrt{1+c^2})^{-1} & (c>0) \\ -c-\sqrt{1+c^2} = -(|c|+\sqrt{1+c^2})^{-1} & (c<0) \end{cases}$$

由此得

$$\tan\theta=\frac{\mathrm{sign}(c)}{|c|+\sqrt{1+c^2}}$$

$$\cos\theta=\frac{1}{\sqrt{1+\tan^2\theta}}$$

$$\sin\theta=\tan\theta\cos\theta$$

4° 对A作变换$P(i,j)AP^T(i, j)$:

计算经过旋转改变的矩阵元素

$$a_{ik}:=a_{ik}\cos\theta+a_{jk}\sin\theta \qquad (k\neq i,j)$$

$$a_{ki}:=a_{ik}$$

$$a_{jk}:=a_{jk}\cos\theta-a_{ik}\sin\theta \qquad (k\neq i,j)$$

$$a_{kj}:=a_{jk}$$

$$a_{ii}:=a_{ii}\cos^2\theta+a_{jj}\sin^2\theta+2a_{ij}\sin\theta\cos\theta$$

$$a_{jj}:=a_{ii}\sin^2\theta+a_{jj}\cos^2\theta-2a_{ij}\sin\theta\cos\theta$$

$$a_{ij}:=0$$

$$a_{ji}:=0$$

5° 计算$R:=RP^T(i,j)$

实际上，只需计算R的两列元素

$$\begin{cases} r_{ki}:=r_{ki}\cos\theta+r_{kj}\sin\theta \\ r_{kj}:=-r_{ki}\sin\theta+r_{kj}\cos\theta \end{cases} \qquad (k=1,2,\cdots,n)$$

6° 判断$S(A)=\sum\limits_{l\neq k}(a_{lk})^2<\varepsilon$是否满足，其中$\varepsilon$为预先给定的误差量．若不满足，则重复步骤2°～6°直到满足为止．最后，A的对角线元素就是原始矩阵A的特征值，数组R的每一列向量是A的近似的特征向量．

Jacobi方法的计算过程是稳定的，精确度较高，但计算工作量较大，矩阵的稀疏带状将被破坏。

8.7.3 例 用Jacobi方法计算对称矩阵

$$A=\begin{pmatrix} 2 & -1 & 1 \\ -1 & 2 & -1 \\ 1 & -1 & 2 \end{pmatrix}$$

的特征值.

解 1° 选主元素$a_{12}=-1$,则$i=1,j=2$.因为$a_{11}=a_{22}$,所以取$\theta=\frac{\pi}{4}\operatorname{sign}(a_{12})=-\frac{\pi}{4}$,则

$$\sin\theta=-\frac{\sqrt{2}}{2},\quad \cos\theta=\frac{\sqrt{2}}{2}$$

$$A_1=P_1AP_1^T=\begin{pmatrix} \overset{\blacktriangle}{3} & \overset{\blacktriangle}{0} & \underline{0.707107} \\ 0 & 1 & -0.707107 \\ 0.707107 & -0.707107 & 2 \end{pmatrix}$$

2° 主元素$a_{13}^{(1)}=0.707107$,则$i=1,j=3$.因为$a_{11}\neq a_{33}$,计算$c=\frac{a_{11}^{(1)}-a_{33}^{(1)}}{2a_{13}^{(1)}}=0.707107>0$,

则$\tan\theta=-c+\sqrt{1+c^2}=0.517638$,

$\cos\theta=0.888074$, $\sin\theta=0.459701$

$$A_2=P_2A_1P_2^T=\begin{pmatrix} \overset{\blacktriangle}{3.366027} & -0.325058 & \overset{\blacktriangle}{0} \\ -0.325058 & 1 & \underline{-0.627963} \\ 0 & -0.627963 & 1.633975 \end{pmatrix}$$

3° 主元素$a_{23}^{(2)}=-0.627963$,则$i=2,j=3$.因为$a_{22}\neq a_{33}$,计算$c=0.504787$,则$\tan\theta=0.615396$, $\cos\theta=0.851654$, $\sin\theta=0.524104$

$$A_3=\begin{pmatrix} 3.366027 & \overset{\blacktriangle}{\underline{-0.276837}} & \overset{\blacktriangle}{0.170364} \\ -0.276837 & 0.613554 & 0 \\ 0.170364 & 0 & 2.020420 \end{pmatrix}$$

4° 主元素$a_{12}^{(3)}=-0.276837$,则$i=1, j=2$.因为$a_{11}\neq a_{22}$,计算$c=-4.971288<0$,则$\tan\theta=-(c+\sqrt{1+c^2})=-0.0995802$,$\cos\theta=0.995078$,$\sin\theta=-0.0990901$,

$$A_4=\begin{pmatrix} \overset{\blacktriangle}{3.393592} & \overset{\blacktriangle}{0} & \underline{0.169525} \\ 0 & 0.585986 & 0.0168814 \\ 0.169525 & 0.0168814 & 2.020420 \end{pmatrix}$$

5° 主元素$a_{13}^{(4)}=0.169525$,则$i=1, j=3$.计算$c=4.050058>0$,则$\tan\theta=0.121629$,$\cos\theta=0.992684$,$\sin\theta=0.20739$,

$$A_5=\begin{pmatrix} 3.414209 & 0.00203824 & 0 \\ 0.00203824 & 0.585986 & 0.0167579 \\ 0 & 0.0167579 & 1.999800 \end{pmatrix}$$

$S(A_5)=0.00028498$,于是A的特征值为

$\lambda_1\approx3.414209$,$\lambda_2\approx0.585986$,$\lambda_3\approx1.999800$.

A的精确的特征值为

$$\lambda_1=2\left(1+\frac{\sqrt{2}}{2}\right)\approx3.414214,$$

$$\lambda_2=2\left(1-\frac{\sqrt{2}}{2}\right)\approx0.585786,\quad \lambda_3=2$$

再求出$R_5^T=P_1^TP_2^TP_3^TP_4^TP_5^T$的列向量,即得$A$的近似特征向量.

8.7.2 循环Jacobi法

循环Jacobi法是Jacobi算法的变形,它不需要像Jacobi法那样去寻求最大模的元素,而是按照一定的循环次序把矩阵的非对角元素消去.比如,按行循环消去法,就是按照下列各行的次序

$$
\begin{array}{llll}
(1,2) & (1,3) & \cdots & (1,n) \\
 & (2,3) & \cdots & (2,n) \\
 & & & \cdots \\
 & & & (n-1,n)
\end{array}
$$

选取旋转平面依次消去矩阵上三角部分的 (i,j)元素。如果 $|a_{ij}^{(m)}| \ll D(A_m)$ (A_m的对角线元素的平方和)，则可以跳过这一步，从(1,2) 到$(n-1, n)$称为一轮。

8.7.3 Jacobi过关法

Jacobi过关法也是 Jacobi 算法的变形，它不去寻找最大模元素，而采用如下过关办法：

计算对称矩阵A的非对角元素之平方和$S(A)$

$$
v_0 = (2\sum_{l=1}^{n-1}\sum_{k=l+1}^{n} a_{lk}^2)^{\frac{1}{2}} = [S(A)]^{\frac{1}{2}}
$$

1° 设置关口 (threshold)$v_1 = \frac{v_0}{n}$，在矩阵A 的非对角线元素

$$
\begin{array}{l}
a_{12}, a_{13}, \cdots, a_{1n} \\
\quad a_{23}, \cdots, a_{2n} \\
\qquad \vdots \\
\qquad a_{n-1,n}
\end{array}
$$

中，按行（或按列）扫描，即按次序$a_{12}, a_{13}, \cdots, a_{1n}, a_{23}, \cdots, a_{2n}, \cdots, a_{n,n-1}$ (或$a_{12}, a_{13}, a_{23}, \cdots, a_{1n}, a_{2n}, \cdots a_{n,n-1}$)。

逐次比较。当非对角元素

$|a_{ij}| \geqslant v_1$

时，则选择形如(8.7.1) 的平面旋转矩阵$p(i,j)$，使 a_{ij} 化为零；否则让元素a_{ij}过关。由于在某次消除了的元素，可能在下一轮中又重新增长，因此要经过多次扫描，一直约化到$A_m = (a_{ik}^{(m)})_{n\times n}$的所有非对角元素都满足 $|a_{ij}^{(m)}| < v_1$为止。

2° 缩小关口，即$\nu_2=\frac{\nu_1}{n}$.对A_m重复1°的步骤，经过多轮扫描一直约化到A_r的非对角元素都满足

$|a_{ij}^{(r)}|<\nu_2$

3° 再缩小关口.重复上述过程，经过一系列的关口

$$\nu_i=\frac{\nu_{i-1}}{n} \qquad (i=1,2,\cdots)$$

直到$\nu_i\leqslant\left(\frac{\varepsilon}{n}\right)\nu_0$为止，其中$\varepsilon$为给定的精度要求.1°—3°中的$n$是矩阵$A$的阶数.

8.8

8.8 Givens方法和Householder方法

用Givens（吉文斯）方法和Householder（豪斯荷尔德）方法可以把实矩阵$A\in\boldsymbol{R}^{n\times n}$化简，它们都是对$A$作**正交相似变换**（Orthogonal Similarity Transformation）.正交变换不会使非对称矩阵特征值的稳定性变坏，正交矩阵有较好的误差传播性质.有二种简化情况：

1° 用正交相似变换约化一般实矩阵为上Hessenberg（赫申伯格）阵，如矩阵$B=(b_{ij})$，

8.8.1
$$B=\begin{pmatrix}\times&\times&\times&\times&\times\\\times&\times&\times&\times&\times\\&\times&\times&\times&\times\\&&\times&\times&\times\\&&&\times&\times\end{pmatrix}$$

当$i>j+1$时有$b_{ij}=0$，则称B为上Hessenberg阵.

2° 用正交相似变换约化实对称矩阵为三对角阵.

这样，求原矩阵A的特征值问题，就转化为求上Hessenberg阵或求对称三对角阵的特征值问题.

这类算法可以在有限步中完成，这是一个重要特征，是Jacobi算法所不能及的，因为Jacobi方法在对角化的每个阶段(8.7.2)中，都必须对整个矩阵进行运算，而且在某步已被化为零的元素，在以后各步中可能变为非零且是不能忽略的.特别对大型矩阵，Jacobi方法可能是一个收敛缓慢的无限过程.

8.8.1 Givens方法

设A是n阶矩阵，$P(j,k)$是x_j, y_k平面内的Givens平面旋转矩阵

$$P(j,k)=\begin{pmatrix} 1 & & & & & & & & \\ & \ddots & & & & & & & \\ & & 1 & & & & & & \\ & & & \cos\theta & & & \sin\theta & & \\ & & & & 1 & & & & \\ & & & & & \ddots & & & \\ & & & & & & 1 & & \\ & & & -\sin\theta & & & \cos\theta & & \\ & & & & & & & 1 & \\ & & & & & & & & \ddots \\ & & & & & & & & & 1 \end{pmatrix} \begin{matrix} \\ \\ \\ j \\ \\ \\ \\ k \\ \\ \\ \\ \end{matrix}$$

是正交矩阵，$P^TP=I_n$.

选择旋转角使$P(j+1,k)$左乘A_s能消去元素$a_{kj}^{(s)}$. 计算$a_{kj}^{(s+1)}$的一般公式是

8.8.2 $a_{kj}^{(s+1)}=-a_{j+1,j}^{(s)}\sin\theta+a_{kj}^{(s)}\cos\theta$

其中s表示消元步数.

如果选取

8.8.3 $\tan\theta=\dfrac{a_{kj}^{(s)}}{a_{j+1,j}^{(s)}}$

则$a_{kj}^{(s+1)}$将为零.

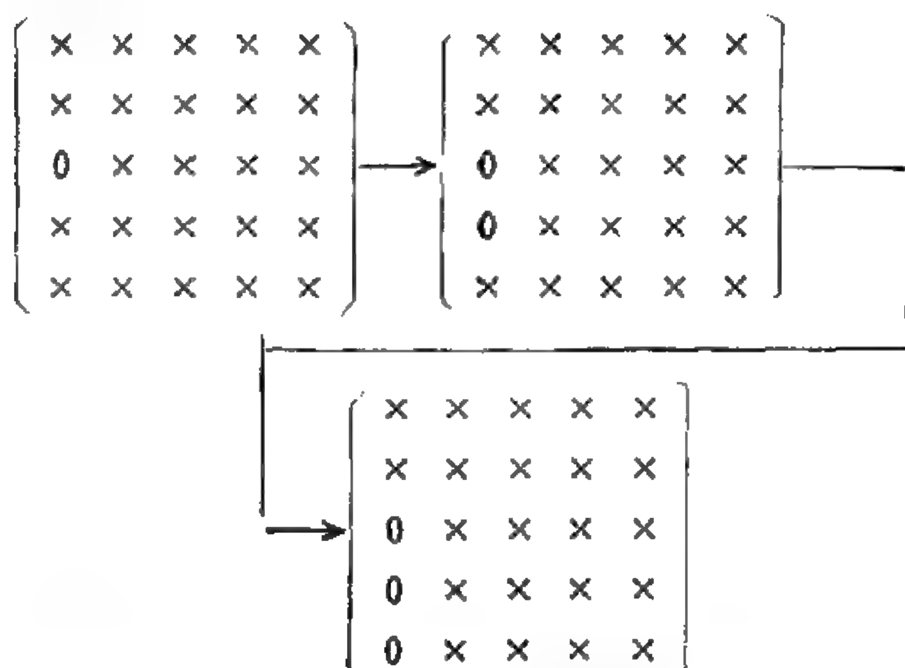
再用$P^T(j+1,k)$右乘矩阵A_s，相当于对A_s作了一次相似变换，

8.8.4 $A_{s+1}=P(j+1,k)A_sP^T(j+1,k)$

变换后，当A为实矩阵时，则$a_{kj}^{(i+1)}=0$；当A为实对称矩阵时，则$a_{kj}^{(i+1)}=a_{jk}^{(i+1)}=0$。

可以按照特殊的次序来选择这种旋转平面，以使得用某个平面旋转已经被化为零的元素在以后的变换中不再变为非零元素。以五阶矩阵为例，旋转的正确排列顺序如下：从矩阵的第1列开始，用$P(2,3)$，$P(2,4)$，$P(2,5)$的变换分别消去(3,1)，(4,1)，(5,1)位置上的元素，如图（8.8.5）所示：

8.8.5 图



然后用$P(3,4)$，$P(3,5)$的变换分别消去（4,2），(5,2)位置上的元素，用$P(4,5)$变换消去(5,3)上的元素，再把五阶实矩阵A约化为上Hessenberg阵。如果A对称，则在作$P(2,3)$，…，$P(4,5)$变换时，把(1,3)，(1,4)，(1,5)，(2,4)，(2,5)(3,5)位置上的元素消去，最后把A约化为对称三对角阵。

按照上列顺序作变换，能使某一变换变成零的元素在以后的变换中不受影响，而且总的变换次数M是有限的

$$M\leqslant\frac{1}{2}(n-1)(n-2)$$

其中n为矩阵的阶数。如五阶矩阵的$M=6$。

8.8.6 Givens方法的计算步骤

1° 对于$j=1,2,\cdots,n-3$，执行2°；

2° 对于$k=j+2,\cdots,n$，执行下列各式：

$$\tan\theta=\frac{a_{k,j}^{(j)}}{a_{j+1,j}^{(j)}}$$

当$a_{j+1,j}^{(j)}=0$时跳过这一步，做下一个k，否则

$$\cos\theta=\frac{1}{\sqrt{1+\tan^2\theta}}$$

$$\sin\theta=\tan\theta\cos\theta$$

$$A_{j+1}=P(j+1,k)A_jP^T(j+1,k)$$

8.8.7 例 应用Givens方法把矩阵

$$A=\begin{pmatrix}1 & 2 & 1 & 2\\ 2 & 2 & -1 & 1\\ 1 & -1 & 1 & 1\\ 2 & 1 & 1 & 1\end{pmatrix}$$

化为三对角阵。

解 令$A_1=A$。

1° $j=1$，$k=3$，$\tan\theta=\dfrac{a_{31}}{a_{21}}=0.5$

$$P(2,3)=\begin{pmatrix}1 & 0 & 0 & 0\\ 0 & 0.8944 & 0.4472 & 0\\ 0 & -0.4472 & 0.8944 & 0\\ 0 & 0 & 0 & 1\end{pmatrix}$$

$$A_2=P(2,3)A_1P^T(2,3)$$

$$=\begin{pmatrix}1 & 2.2361 & 0 & 2\\ 2.2361 & 1 & -1 & 1.3416\\ 0 & -1 & 2 & 0.4472\\ 2 & 1.3416 & 0.4472 & 1\end{pmatrix}$$

$2°\quad j=1,\ k=4,\ \tan\theta=\frac{a_{41}}{a_{21}}=0.8944$

$$A_3=\begin{pmatrix}1 & 3 & 0 & 0\\ 3 & 2.3333 & -0.4472 & 0.1491\\ 0 & -0.4472 & 2 & 1\\ 0 & 0.1491 & 1 & -0.3333\end{pmatrix}$$

$3°\quad j=2,\ k=4,\ \tan\theta=\frac{a_{42}}{a_{32}}=-0.3333$

$$A_4=\begin{pmatrix}1 & 3 & 0 & 0\\ 3 & 2.3333 & -0.4714 & 0\\ 0 & -0.4714 & 1.1667 & 1.5000\\ 0 & 0 & 1.5000 & 0.5000\end{pmatrix}$$

8.8.2 Householder方法

Householder（豪斯荷尔德）方法是用对称正交阵

8.8.8 $H^{(r)}=I-2w^{(r)}(w^{(r)})^T$

其中$w^{(r)}=(0,0,\cdots,0,\omega_{r+1}^{(r)},\cdots\omega_n^{(r)})^T$

$(w^{(r)})^Tw^{(r)}=1$

对矩阵$A\in R^{n\times n}$作正交相似变换，将实矩阵A约化为上Hessenberg阵(8.8.1)；当A对称时，将A约化为对称三对角阵.$H^{(r)}$是对称阵($H^{(r)T}=H^{(r)}$)、正交阵($H^{(r)T}H^{(r)}=I_n$)和对合阵($H^{(r)2}=I_n$).

令$A_1=A$，可以选择$H^{(r)}$，使得对A_1作变换$A_2=H^{(1)}A_1H^{(1)}$后，位于A_2第1列的(3,1),(4,1),…,(n,1)处的元素为零，使$A_3=H^{(2)}A_2H^{(2)}$的位于第2列(4,2),(5,2)…(n,2)处的元素为零；直到$A_n=H^{(n-1)}A_{n-1}H^{(n-1)}$为上Hessenberg阵止.选择的$H^{(r)}$需使在某次变换已被消成零的元素处

不再产生非零元素.

为了选择$H^{(r)}$，令x是矩阵A_r的第r列，将x分块如下：

$x^T=(\hat{x}^T, x_{r+1}, \hat{u}^T)$

其中$\hat{x}$是x的前r个元素组成的列向量，$\hat{u}$是x的后$n-(r+1)$个元素组成的列向量.与此对应，令

$w^{(r)T}=(0,\omega_{r+1}^{(r)}, v^r)$

则有

$$H^{(r)}x=(I-2w^{(r)}w^{(r)})^T x$$

$$=\begin{pmatrix}\hat{x}\\ x_{r+1}\\ \hat{u}\end{pmatrix}-2\begin{pmatrix}0\\ \omega_{r+1}^{(r)}\\ v\end{pmatrix}(0,\omega_{r+1}^{(r)}, v^T)\begin{pmatrix}\hat{x}\\ x_{r+1}\\ \hat{u}\end{pmatrix}=\begin{pmatrix}\hat{x}\\ x_{r+1}-2z\omega_{r+1}^{(r)}\\ \hat{u}-2zv\end{pmatrix}$$

其中

$z=(\omega_{r+1}^{(r)}x_{r+1}+\hat{u}^T\hat{u})$

使$H^{(r)}x=(\hat{x}^T,a,o^r)^T$以及$(w^{(r)})^Tw^{(r)}=1$.因而必须选择$v=\alpha\hat{u}$，$\omega_{r+1}^{(r)}=\alpha u_{r+1}^{(r)}$，这样得到

8.8.9 $1-\alpha^2[2u_{r+1}^{(r)}x_{r+1}+2\hat{u}^T\hat{u}]=0$

等价于$1-2\alpha z=0$；

8.8.10 $1-\alpha^2[(u_{r+1}^{(r)})^2+\hat{u}^T\hat{u}]=0$

等价于$(w^{(r)})^Tw^{(r)}=1$.

从(8.8.9)和(8.8.10)解出α和u_{r+1}，最后有

$u_{r+1}=x_{r+1}\pm\{(x_{r+1})^2+\hat{u}^T\hat{u}\}^{\frac{1}{2}}=x_{r+1}\pm\sigma_r$

8.8.11 $\alpha^2=(2\sigma_r(x_{r+1}+\sigma_r))^{-1}$

为了避免正负相消，取σ_r与x_{r+1}的符号一致，由此得

8.8.12 $w^{(r)}=\alpha(0,x_{r+1}+\sigma_r, \hat{u}^T)^T=\alpha(0,u_r^T)^T$,

$u_r^T=(x_{r+1}+\sigma_r,\ \hat{u}^T)$

8.8.13 $\alpha=-\sigma_r=-\{x_{r+1}^2+\hat{u}^T\ \hat{u}\}^{\frac{1}{2}}=-\|u\|_2$

8.8.14 $H^{(r)}=I_n-2w^{(r)}(w^{(r)})^T=\begin{pmatrix} I_r & O \\ O & R_r \end{pmatrix}$

其中

8.8.15 $R_r=I_{n-r}-2\alpha^2 u_r u_r^T=I_{n-r}-\dfrac{u_r u_r^T}{\beta_r}$

$\beta_r=(2\alpha^2)^{-1}=\sigma_r(u_{r+1}+\sigma_r)$

用Householder 法正交相似约化一般矩阵和对称矩阵的步骤如下：

1° 将矩阵A进行分块

$$A=\left(\begin{array}{c|ccc} a_{11} & a_{12}, & \cdots, & a_{1n} \\ \hline a_{21} & a_{22} & & a_{2n} \\ \vdots & & & \\ a_{n1} & a_{n2} & & a_{nn} \end{array}\right)=\begin{pmatrix} a_{11} & A_{12}^{(1)} \\ a_{21} & A_{22}^{(1)} \end{pmatrix}$$

设$a_{21}^{(1)}\neq 0$，否则不需要约化，选择

$$H^{(1)}=\begin{pmatrix} 1 & 0 \\ 0 & R_1 \end{pmatrix}$$

使$H^{(1)}\begin{pmatrix} a_{11} \\ a_{21} \end{pmatrix}=\begin{pmatrix} a_{11} \\ R_1 a_{21} \end{pmatrix}=\begin{pmatrix} a_{11} \\ -\sigma_1 e_1 \end{pmatrix}$

其中$e_1=(1,0,\cdots,0)^T$是$n-1$维向量。

由（8.8.15）有

$$\begin{cases} R_1=I_{n-1}-\beta_1^{-1}u_1 u_1^T \\ u_1=a_{21}^{(1)}+\sigma_1 e_1=(a_{21}+\sigma_1, a_{31}, \cdots, a_{n1})^T \\ \sigma_1=\text{sign}(a_{21})\left(\sum\limits_{i=2}^{n} a_{i1}^2\right)^{\frac{1}{2}} \\ \beta_1=\sigma_1(\sigma_1+a_{21}) \end{cases}$$

则

$$A_2=H^{(1)}A_1H^{(1)}=\begin{pmatrix} a_{11} & A_{12}^{(1)}R_1 \\ R_1\boldsymbol{a}_{21} & R_1A_{22}^{(1)}R_1 \end{pmatrix}$$

$$=\left(\begin{array}{c|c|ccc} a_{11} & a_{12}^{(2)} & a_{13}^{(2)} & \cdots & a_{1n}^{(2)} \\ -\sigma_1 & a_{22}^{(2)} & a_{23}^{(2)} & \cdots & a_{2n}^{(2)} \\ \hline & a_{32}^{(2)} & a_{33}^{(2)} & \cdots & a_{3n}^{(2)} \\ & \vdots & & & \\ & a_{n2}^{(2)} & a_{n3}^{(2)} & \cdots & a_{nn}^{(2)} \end{array}\right)$$

$$=\begin{pmatrix} A_{11}^{(2)} & \boldsymbol{a}_{12}^{(2)} & A_{13}^{(2)} \\ & \boldsymbol{a}_{22}^{(2)} & A_{23}^{(2)} \end{pmatrix}$$

2° 设对A已进行了第$r-1$步正交相似变换，即

$$A_r=H^{(r-1)}A_{r-1}H^{(r-1)}$$

$$=\left(\begin{array}{cccc|c|ccc} a_{11} & a_{12}^{(2)} & \times\cdots\times & & a_{1r}^{(r)} & a_{1,r+1}^{(r)} & \cdots & a_{1n}^{(n)} \\ -\sigma_1 & a_{22}^{(2)} & & & a_{2r}^{(r)} & \cdots & \cdots & \cdots \\ & \ddots & & & \vdots & & & \\ & & & -\sigma_{r-1} & a_{rr}^{(r)} & a_{r,r+1}^{(r)} & \cdots & a_{r,n}^{(r)} \\ \hline & & & & a_{r+1,r}^{(r)} & a_{r+1,r+1}^{(r)} & \cdots & a_{r+1,n}^{(r)} \\ & & & & \vdots & \cdots & \cdots & \cdots \\ & & & & a_{nr}^{(r)} & a_{n,r+1}^{(r)} & \cdots & a_{nn}^{(r)} \end{array}\right)$$

$$=\begin{pmatrix} A_{11}^{(r)} & \boldsymbol{a}_{12}^{(r)} & A_{13}^{(r)} \\ 0 & \boldsymbol{a}_{22}^{(r)} & A_{23}^{(r)} \end{pmatrix}$$

选择

$$H^{(r)}=\begin{pmatrix} I_r & 0 \\ 0 & R_r \end{pmatrix}$$

使

$$H^{(r)}\begin{pmatrix} \boldsymbol{a}_{12}^{(r)} \\ \boldsymbol{a}_{22}^{(r)} \end{pmatrix}=\begin{pmatrix} \boldsymbol{a}_{12}^{(r)} \\ R_r\boldsymbol{a}_{22}^{(r)} \end{pmatrix}=\begin{pmatrix} \boldsymbol{a}_{12}^{(r)} \\ -\sigma_r\boldsymbol{e}_1 \end{pmatrix}$$

由于

$$\begin{cases} R_r = I_{n-r} - \beta_r^{-1} \boldsymbol{u}_r \boldsymbol{u}_r^T \\ \boldsymbol{u}_r = \boldsymbol{a}_{21}^{(r)} + \sigma_r \boldsymbol{e}_1 \\ \sigma_r = \operatorname{sign}(a_{r+1,r}^{(r)}) \left(\sum_{i=r+1}^{n} a_{ir}^{(r)2} \right)^{\frac{1}{2}} \\ \beta_r = \sigma_r (\sigma_r + a_{r+1,r}^{(r)}) \end{cases}$$

则 $A_{r+1} = H^{(r)} A_r H^{(r)} = \begin{pmatrix} A_{11}^{(r)} & \boldsymbol{a}_{12}^{(r)} & A_{12}^{(r)} R_r \\ 0 & R_r \boldsymbol{a}_{21}^{(r)} & R_r A_{22}^{(r)} R_r \end{pmatrix}$

$$= \begin{pmatrix} A_{11}^{(r)} & \boldsymbol{a}_{12}^{(r)} & A_{12}^{(r)} R_r \\ 0 & -\sigma_r \boldsymbol{e}_1 & R_r A_{22}^{(r)} R_r \end{pmatrix}$$

重复2°直到约化成上Hessenberg阵为止，共约化$n-2$步，

$H^{(n-2)} \cdots H^{(2)} H^{(1)} A H^{(1)} H^{(2)} \cdots H^{(n-2)}$

$$= A_{n-1} = \begin{pmatrix} a_{11} & \times & \times & \cdots & \times \\ -\sigma_1 & a_{22}^{(1)} & \times & \cdots & \times \\ & -\sigma_2 & a_{33}^{(2)} & \cdots & \times \\ & & \ddots & \ddots & \vdots \\ & & & -\sigma_{n-1} & a_{nn}^{(n-1)} \end{pmatrix}$$

如果$A \in R^{n \times n}$为对称矩阵，则A_{n-1}为对称三对角阵。

8.8.16 例 应用Householder方法把矩阵

$$A = \begin{pmatrix} 1 & 2 & 1 & 2 \\ 2 & 2 & -1 & 1 \\ 1 & -1 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{pmatrix}$$

化为三对角矩阵。

解 1° 把A分块

$$A_1 = A = \left(\begin{array}{c|ccc} 1 & 2 & 1 & 2 \\ \hline 2 & 2 & -1 & 2 \\ 1 & -1 & 1 & 1 \\ 2 & 1 & 1 & 1 \end{array}\right) = \begin{pmatrix} a_{11} & A_{12}^{(1)} \\ \boldsymbol{a}_{21}^{(1)} & A_{22}^{(1)} \end{pmatrix}$$

$$\boldsymbol{a}_{21}^{(1)}=(2,1,2)^T$$

$$\sigma_1=(2^2+1^2+2^2)^{\frac{1}{2}}=3$$

$$\boldsymbol{u}_1=\boldsymbol{a}_{21}^{(1)}+\sigma_1\boldsymbol{e}_1=(5,1,2)^T$$

$$\beta_1=\sigma_1(a_{21}+\sigma_1)=3(2+3)=15$$

$$R_1=I-\frac{1}{\beta_1}\boldsymbol{u}_1\boldsymbol{u}_1^T=\frac{1}{15}\begin{pmatrix}-10 & -5 & -10\\ -5 & 14 & -2\\ -10 & -2 & 11\end{pmatrix}$$

$$H^{(1)}=\begin{pmatrix}1 & 0\\ 0 & R_1\end{pmatrix}$$

$$A_2=H^{(1)}A_1H^{(1)}=\left(\begin{array}{c|c|cc}1 & -3 & 0 & 0\\ -3 & \frac{7}{3} & \frac{7}{15} & -\frac{1}{15}\\ \hline 0 & \frac{7}{15} & \frac{118}{75} & \frac{101}{75}\\ 0 & -\frac{1}{15} & \frac{101}{75} & \frac{7}{75}\end{array}\right)$$

2° 把A_2分块如上式

$$\boldsymbol{a}_{22}^{(2)}=(\frac{7}{15},-\frac{1}{15})^T$$

$$\sigma_2=\left[(\frac{7}{15})^2+(\frac{1}{15})^2\right]^{\frac{1}{2}}=\frac{\sqrt{50}}{15}=0.4714$$

$$\boldsymbol{u}_2=\boldsymbol{a}_{22}^{(2)}+\sigma_2\boldsymbol{e}_1=(\frac{7}{15}+\frac{\sqrt{50}}{15},-\frac{1}{15})^T$$

$$=(0.9381,-0.0667)^T$$

$$\beta_2=\sigma_2(\frac{7+\sqrt{50}}{15})=\frac{\sqrt{50}(7+\sqrt{50})}{15\times 15}=0.4422$$

$$R_2=I-\frac{1}{\beta_2}\boldsymbol{u}_2\boldsymbol{u}_2^T$$

$$H^{(1)}=\begin{pmatrix} I_2 & 0 \\ 0 & R_2 \end{pmatrix}$$

$$A_3=\begin{pmatrix} 1 & -3 & 0 & 0 \\ -3 & 2.3333 & -0.4714 & 0 \\ 0 & -0.4714 & 1.1667 & -1.5000 \\ 0 & 0 & -1.5000 & 0.5000 \end{pmatrix}$$

8.9 用二分法求实对称三对角矩阵的特征值

Givens（吉文斯）方法和Householder（豪斯荷尔德）方法是把实对称矩阵变换成对称三对角阵

8.9.1
$$T=\begin{pmatrix} d_1 & e_1 & & & & \\ e_1 & d_2 & e_2 & & & \\ & e_2 & d_3 & e_3 & & \\ & & \ddots & \ddots & \ddots & \\ & & & e_{n-2} & d_{n-1} & e_{n-1} \\ & & & & e_{n-1} & d_n \end{pmatrix}$$

可以假定e_i均不为零，否则可以把T分为较低阶的三对角矩阵。

如果要求出T的全部特征值和特征向量，那末(8.10)的QR算法是最有效的算法。但是，如果求T的某一个特征值或者求某个区间内的若干个特征值，则利用Sturm（斯图姆）序列的特性很容易将矩阵T的特征值分离。T的特征值一经分离，则可用二分法较快地求出，然后用反幂法有效地计算其相应的特征向量。

8.9.2 定理 Gerschgorin（格尔施戈林）圆盘定理

设$A=(a_{ij})_n$，则A的每一个特征值必属于下述某个圆盘之中，

$$|\lambda-a_{ii}|\leqslant\sum_{\substack{j=1\\ j\neq i}}^{n}|a_{ij}|\quad(i=1,2,\cdots,n)$$

这说明定理中 n 个不等式确定复平面上的 n 个圆盘，A 的每一个特征值至少在一个圆盘中.

8.9.1 Sturm序列

(8.9.1) 中对称三对角阵 T 的特征多项式是

8.9.3 $f_n(\lambda)=\det(T-\lambda I_n)$

令 $f_1(\lambda),f_2(\lambda),\cdots,f_n(\lambda)$ 表示 $T-\lambda I_n$ 的顺序主子式，即

8.9.4
$$f_k(\lambda)=\begin{vmatrix} d_1-\lambda & e_1 & & & \\ e_1 & d_2-\lambda & e_2 & & \\ & \ddots & \ddots & \ddots & \\ & & e_{k-2} & d_{k-1}-\lambda & e_{k-1} \\ & & & e_{k-1} & d_k-\lambda \end{vmatrix}$$

$$(k=1,2,\cdots,n)$$

规定 $f_0(\lambda)\equiv1$，则可以从 (8.9.4) 得出下列关系式

$$f_1(\lambda)=d_1-\lambda$$

$$f_k(\lambda)=(d_k-\lambda)f_{k-1}(\lambda)-e_{k-1}^2f_{k-2}(\lambda)$$

$$(k=2,3,\cdots,n)$$

序列 $\{f_0(\lambda),f_1(\lambda),\cdots f_n(\lambda)\}$ 是一个Sturm序列，它有一些特殊性质. 为了说明这些性质，用整数函数 $a(\lambda)$ 表示序列 $\{f_k(\lambda)\}$ 中相邻两项符号一致的数目. 例如 $n=3$，$\{f_0(\lambda),f_1(\lambda),f_2(\lambda),f_3(\lambda)\}=\{2,3,-1,1\}$，其符号为 $++-+$ 给出 $a(\lambda)=1$，如果 $f_k(\lambda)$ 为零，那么其符号取 $f_{k-1}(\lambda)$ 的符号。

8.9.5 **定理/Sturm 序列的性质** 序列 $\{f_k(\lambda)\}$ 中相邻两项符号一致的项数 $a(a)$ 等于 T 的严格大于 a 的根的数目. 设 $a<b$，则 T 在 $(a,b]$ 内根的数目等于 $a(a)-a(b)$.

这样，如果 $a(a)=k_1+1,a(b)=k_1$，在区间 $(a,b]$ 内则只有一个特征值.

8.9.6 **例** 矩阵

$$T=\begin{bmatrix}-2 & 1 & & \\ 1 & -2 & 1 & \\ & 1 & -2 & 1 \\ & & 1 & -2\end{bmatrix}$$

是否负定的？在区间[−2，0]内有多少个特征根？

解 令$\lambda=0$，那么

$f_0(0)=1$

$f_1(0)=-2$

$f_2(0)=(-2)(-2)-1=3$

$f_3(0)=(-2)3-1(-2)=-4$

$f_4(0)=(-2)(-4)-1(3)=5$

由于序列$\{f_k(0)\}$的符号是交替的，所以$a(0)=0$。由此可知，T是负定的。用$\lambda=-2$，得到序列

$f_0(-2)=1$

$f_1(-2)=-2+2=0$

$f_2(-2)=0-1=-1$

$f_3(-2)=0-0=0$

$f_4(-2)=0+1=1$

这时，存在两个具有零值的$f_k(-2)$，其符号与$f_{k-1}(-2)$相同，符号序列定为

$\{+,+,-,-,+\}$

所以$a(-2)=2$，T在[−2，0]内的特征值个数等于$a(-2)-a(0)=2$。

8.9.2 二分法

利用Sturm序列的性质及二分法，便能按规定的精度去确定任何特征值。

在只含一个特征值的区间$[l^{(0)},u^{(0)}]$内求特征值λ，可利用

Sturm 序列。如果找到 $l^{(0)}<u^{(0)}$，使得 $\alpha(l^{(0)})=m+1$，$\alpha(u^{(0)})=m(m<n)$，则区间$[l^{(0)}, u^{(0)}]$内仅有一个特征值 λ。令$\lambda^{(0)}=\frac{1}{2}(l^{(0)}+u^{(0)})$，计算 $\alpha(\lambda^{(0)})$，其值只能是 m 或 $m+1$，如果 $\alpha(\lambda^{(0)})=m$，那么 λ 位于区间$[l, \lambda^{(0)}]$内；否则它在$[\lambda^{(0)}, u]$内。反复应用这个方法，可确定 λ 的一个收緊区间 (Tight Interval)，这就是二分法 (Bisection)。其计算步骤如下：

1° 计算 $\lambda^{(k)}=\frac{1}{2}[l^{(k)}+u^{(k)}]$；

2° 计算 $\alpha(\lambda^{(k)})$；

3° 如果 $\alpha(\lambda^{(k)})=\alpha(l^{(k)})$，则

$l^{(k+1)}=\lambda^{(k)}, u^{(k+1)}=u^{(k)}$

否则 $l^{(k+1)}=l^{(k)}$，$u^{(k+1)}=\lambda^{(k)}$；

4° 判断 $|l^{(k+1)}-u^{(k+1)}|<\varepsilon$是否成立，$\varepsilon$为给定的精度，如果不成立，则返回到 1°；若成立，则 $\lambda^{(k+1)}=\frac{1}{2}[l^{(k+1)}+u^{(k+1)}]$就是所求的特征值 λ。

对于 (8.9.1) 中矩阵 T，设 T 的全部特征值的排列次序是

$\lambda_1>\lambda_2>\cdots>\lambda_m>\cdots\lambda_{n-1}>\lambda_n$

8.9.7 用二分法计算前 m 个特征值的算法

1° 利用 Gershgorin 定理 (8.9.2) 计算特征值的下界 a 和上界 b

$$a=\min_{1\leqslant i\leqslant n}\{d_i-|e_i|-|e_{i-1}|\}$$

$$b=\max_{1\leqslant i\leqslant n}\{d_i+|e_i|+|e_{i-1}|\}$$

其中 $e_0=e_n=0$

$l:=a$, $u:=b$

2° 送下界：$\lambda_i := a$ $(i=1,2,\cdots m)$

3° 对于 $k=1,2,\cdots,m$，做下面各步：

1) $l:=\lambda_k$；

2) 计算$\lambda=(u+l)/2$；

3) 如果 $u-l<\varepsilon$，则转6)，否则计算$a(\lambda)$；

4) 如果 $a(\lambda)<k$，则 $u:=\lambda$，否则

$$\begin{cases} l:=\lambda \\ \lambda_i:=\lambda \quad (i=k+1,\cdots,\min(a(\lambda),m)) \end{cases}$$

5) 转 2)继续迭代；

6) $\lambda_k:=\lambda$

8.9.8 例 利用二分法计算对称三对角阵

$$T=\begin{pmatrix} 1 & 2 & 0 \\ 2 & -1 & 1 \\ 0 & 1 & 3 \end{pmatrix}$$

的负特征值.

解 写出 Sturm 序列，即

$f_0(\lambda)=1$

$f_1(\lambda)=1-\lambda$

$f_2(\lambda)=-(1+\lambda)f_1(\lambda)-4$

$f_3(\lambda)=(3-\lambda)f_2(\lambda)-f_1\lambda$

1° $a=\min\{1-2,-1-3,3-1\}$

$=\min\{-1,-4,2\}=-4$

$b=\max\{1+2,-1+3,3+1\}$

$=\max\{3,2,4\}=4$

2° $\lambda=(a+b)/2=(-4+4)/2=0$

3° 计算$\{f_k(0)\}=\{1,1,-5,-14\}$，有$a(0)=2$，这说明 $\lambda_1,\lambda_2\in(0,4]$，$\lambda_3\in[-4,0]$，

4° 计算 λ_3：

$\lambda=\frac{-4+0}{2}=-2$

$\{f_k(-2)\}=\{1,3,-1,-8\}$，有$a(-2)=2$，则$\lambda_3\in[-4,-2]$

$\lambda=\frac{-4-2}{2}=-3$

$\{f_k(-3)\}=\{1,4,4,20\}$，有$a(-3)=3$，则$\lambda_3\in[-3,-2]$

$\lambda=\frac{-3-2}{2}=-2.5$

$\{f_k(-2.5)\}=\{1,3.5,1.25,3.375\}$，有$a(-2.5)=3$，
则$\lambda_3\in[-2.5,-2]$

最后，$\lambda_3=\frac{-2.5-2}{2}=-2.25$

8.10

8.10 LR和QR算法

LR和QR算法都是用来求任意矩阵的全部特征值和特征向量的，其基本思想相同，只是采用的分解方法有区别。(6.2) 中的消去法，原则上是将矩阵A分解成下三角阵L与上三角阵U的乘积，即

$A=LU$

其中L是非奇异的。在 (8.8.2) 中，用 Householder 方法对A作分解时，它在原则上将A分解成正交阵Q和上三角阵R之积，即

$A=QR$

其中Q是非奇异的。这两种分解方法是LR法和QR法的基础。

LR和QR算法是构造一个矩阵序列$\{A_k\}$，在一定条件下，A_k趋向于一个块上三角矩阵，对角块是一阶或二阶的。一阶阵包含A的一个特征根，二阶阵包含A的一对共轭复根。

8.10.1 LR算法

令$A_1=A$，首先对A_1作LU三角分解，则有

$A_1=L_1R_1$

其中R_1为上三角阵，以代替U。然后用L_1对A_1作相似变换，则有

$A_2=L_1^{-1}A_1L_1=L_1^{-1}L_1R_1L_1=R_1L_1$

实际上将A_1分解出的两个矩阵次序颠倒过来相乘。最后再对A_2作三角分解，

$A_2=L_2R_2$

迭代地运用这个技巧，便构成了基本的LR算法。

8.10.1 基本的LR算法

由

$$\begin{cases} A_k=L_kR_k & (\text{将}A_k\text{分解为}L_k\text{和}R_k) \\ A_{k+1}=R_kL_k & (\text{将}R_k\text{乘以}L_k\text{得}A_{k+1}) \end{cases} \quad (k=1,2,\cdots)$$

得到矩阵序列$A,A_1,A_2,\cdots A_k,\cdots$,它们都是相似的，并都有相同的特征值。

8.10.2 LR算法的性质

1° 当A为对称正定矩阵时，LR算法收敛于对角形矩阵，其对角线元素是A的特征值。

2° 当A为实矩阵时，把A预先用相似变换化为上Hessenberg（赫申伯格）阵，在LR变换下形状不变。在适当的条件下，A_k趋向于拟上三角形矩阵，其形状为

$$\begin{pmatrix} \boxed{\times} & \times & \times & \times & \times & \times & \cdots & \times \\ & \times & \times & \times & \times & \times & \cdots & \times \\ & \times & \times & \times & \times & \times & \cdots & \times \\ & & & \times & \times & \times & \cdots & \times \\ & & & \times & \times & \times & \cdots & \times \\ & & & & & \boxed{\times} & \cdots & \times \\ & & & & & & \cdots & \boxed{\times} \end{pmatrix} \tag{8.10.3}$$

（其中第2、3行第2、3列及第4、5行第4、5列的2×2子块各以方框框出）

即：它是对角线块矩阵，并且对角线上的块不是1×1就是2×2。显然，求这种形状的矩阵的特征值，只须求出对角线上的块的特征值。

8.10.2 无移位的QR算法

QR算法从理论到实际应用都比较复杂，是求矩阵全部特征值的最有效和应用最广泛的一种方法。QR算法建立在矩阵的正交分解的基础上，即

$A=QR$

其中 Q 是正交阵，R是上三角阵。这种分解是用平面旋转或Householder方法实现的，而且QR算法总是可以实现的。

8.10.4 无移位的QR算法

令$A_1=A$，对A_1作QR分解，则有

$A_1=Q_1R_1$（QR分解）

$A_2=R_1Q_1$

$A_2=Q_2R_2$（QR分解）

…

$A_k=Q_kR_k$（QR分解）

$A_{k+1}=R_kQ_k$

$A_{k+1}=Q_{k+1}R_{j+1}$（QR分解）

…

矩阵序列$\{A_k\}$收敛于形如(8.10.3)的拟上三角矩阵，因为A，$A_1,A_2,\cdots A_k,\cdots$都是相似矩阵，所以最后的拟三角矩阵的特征值就是$A$的特征值。

8.10.5 QR算法的性质

1°如果A的特征值模不相等，记为$|\lambda_1|>|\lambda_2|>\cdots|\lambda_n|$，则QR算法收敛；

2°把A预先用相似变换化为上Hessenberg阵，当A对称时，约化为对称三对角阵，在QR变换下，它们的形状不

变。QR算法是求对称三对角阵特征值的非常有效的算法。

8.10.6 例 用QR法求上Hessenberg阵

$$A=\begin{pmatrix}5 & -2 & -5 & -1\\ 1 & 0 & -3 & 2\\ 0 & 2 & 2 & -3\\ 0 & 0 & 1 & -2\end{pmatrix}$$

的特征值。

解 由QR法可以得到

$$A_{12}=\left(\begin{array}{c|cc|c}4.0000 & 5.0484 & -3.6564 & \cdot\\ \hline 0 & 1.8789 & -3.5910 & \cdot\\ 0 & 1.3290 & -0.1211 & \cdot\\ \hline 0 & 0 & 0 & -1.0000\end{array}\right)$$

由此不难看出，有一个特征值为4，另一个为-1，还有两个应满足

$$\begin{vmatrix}1.87889-\lambda & -3.5910\\ 1.3290 & 0.1211-\lambda\end{vmatrix}=0$$

于是解得$\lambda=1\pm 2i$。

实际上，解矩阵A的特征方程$\lambda^4-5\lambda^3+7\lambda^2-7\lambda^2-7\lambda-20=0$时，它可分解成$(\lambda+1)(\lambda-4)(\lambda^2-2\lambda+5)=0$，故特征值为$-1$，4，$1\pm 2i$，这些与用QR法得到的特征值相同。

8.10.3 具有移位QR算法

无移位的QR算法（8.10.4）的收敛速度，一般不能满足实际应用的需要，为了产生更快的收敛速度，可以用具有移位（Shift）的QR算法。

8.10.7 具有移位的QR算法

$$
\begin{aligned}
&A_k - \alpha_k I = Q_k R_k \ (\text{QR分解}) \\
&A_{k+1} = \alpha_k I + R_k Q_k
\end{aligned}
\qquad (k=1,2,\cdots)
$$

其中 α_k 为移位，$A_1 = A$。

因为

$$R_k = Q_k^T(A_k - \alpha_k I)$$

$$
\begin{aligned}
A_{k+1} &= \alpha_k I + Q_k^T(A_k - \alpha_k I)Q_k = \alpha_k I + Q_k^T A_k Q_k - \alpha_k I \\
&= Q_k^T A_k Q_k \quad (k \geqslant 1)
\end{aligned}
$$

所以 A_{k+1} 与 A_k 相似，$A, A_1, A_2, \cdots A_k, \cdots$ 都有相同的特征值。适当选取位移α_k，可使A_k的最末一行的非对角元素极其迅速地趋于零，而次对角元素也逐渐趋于零。因此，矩阵序列 A_k 收敛于一个分块对角阵，块对角阵不是1×1 的就是 2×2 的，这后一种情况对应于模相等但符号相反的特征值。

移位量 α_k 一般取A_k的一个粗糙的近似特征值，通常有两种选择移位的方法：

1° 选择 $\alpha_k = a_{nn}^{(k)}$，$a_{nn}^{(k)}$ 是矩阵 A_k 右下角的元素。这样选择的移位，虽然能保证收敛，但一般收敛较慢。

2° 当 A 为对称三对角阵时，选择$\alpha_k = \beta_k$，其中β_k是2×2矩阵

$$
\begin{pmatrix}
d_{n-1}^{(k)} & e_n^{(k)} \\
e_n^{(k)} & d_n^{(k)}
\end{pmatrix}
$$

的最接近于 $d_n^{(k)}$ 的特征值。这样选择的α_k产生比1°更快的收敛速度。

当 A 为对称三对角阵（8.9.1）时，上述两种移位方法的任一种都可求得。

$e_n^{(k)} e_{n-1}^{(k)} \to 0$（当 $k \to \infty$时）

如果 $e_n^{(k)}$ 变成可以忽略，则可取 $d_n^{(k)}$ 为特征值，然后删去矩阵A_k的最后一行和最后一列。如果$e_{n-1}^{(k)}$变成可以忽略，而$e_n^{(k)}$ 不可忽略，则可由A_k右下角的2×2矩阵确定两个特征

值，然后删去 A_k 的最后两行和最后两列，依此继续进行下去。这种算法对矩阵的阶数产生一种压缩，因此，收敛速度很快。

8.11　8.11　求实对称三对角矩阵特征值的QL算法

QL 算法是QR算法（8.10）在实对称三对角矩阵情况下的变形，用来求实对称矩阵的全部特征值和特征向量。在使用 QL 算法以前，通过 Householder 正交相似变换把实对称矩阵约化为对称三对角矩阵，再用 QL 算法进行计算。该算法稳定，在实际中经常使用。

设A 是对称三对角矩阵

$$A=\begin{pmatrix} d_1 & e_2 & & & \\ e_2 & d_2 & e_3 & & \\ & \ddots & \ddots & \ddots & \\ & & \ddots & \ddots & e_n \\ & & & e_n & d_n \end{pmatrix}$$

如果对某个 j $(2<j<n)$ 有$e_j=0$，则 A 可化成二个相互无关的低阶矩阵，分别求其特征值；如果 $e_2=0$ 或$e_n=0$，则可立即得到A 的一个特征值 $\lambda_1=d_1$ 或 $\lambda_n=d_n$；除以上二种情况外，可以用 QL 算法。和 QR 算法一样，矩阵的正交分解是 QL 算法的基础，即

8.11.1　$A=QL$

用平面旋转将A 分解成正交矩阵Q 和下三角矩阵 L。

8.11.1　无移位的QL算法

8.11.2　无移位 QL 算法的计算步骤

$A^{(1)}=A=Q^{(1)}L^{(1)}$　(QL分解)

$A^{(2)}=Q^{(1)-1}A^{(1)}Q^{(1)}=L^{(1)}Q^{(1)}$

$A^{(3)}=Q^{(2)}L^{(2)}$ （QL分解）

…

$A^{(k)}=Q^{(k)}L^{(k)}$ （QL分解）

$A^{(k+1)}=L^{(k)}Q^{(k)}$

…

$A^{(k+1)}$是对称三对角矩阵，与A有相同的特征值。当$k\to\infty$时，$A^{(k+1)}$趋向于对角矩阵，A的特征值按数量级由小到大排列在$A^{(k+1)}$的对角线上。

通过一系列的平面旋转$P_n^{(k)}$，$P_{n-1}^{(k)}$，…，$P_2^{(k)}$将$A^{(k)}$分解成$Q^{(k)}$和$L^{(k)}$，其中$P_i^{(k)}$表示在$(i,i+1)$平面上的旋转变换矩阵，用以消去矩阵$(i,i+1)$处的元素，其形式为

$$P_i^{(k)}=\begin{pmatrix} I_{i-2} & & & \\ & c_{i-1} & -s_{i-1} & \\ & s_{i-1} & c_{i-1} & \\ & & & I_{n-i} \end{pmatrix} \tag{8.11.3}$$

旋转过程（以$k=1$为例）如下：

$A_2^{(1)}=P_nA^{(1)}$

使$(n-1,n)$位置上元素值为零，变换的结果使$(n,n-2)$处由零变为非零。接着

$A_3^{(1)}=P_{n-1}A_2^{(1)}$

…

$A_{n-i+2}^{(1)}=P_iA_{n-i+1}^{(1)}$

…

$A_n^{(1)}=P_2A_{n-1}^{(1)}$

其中

$$
A^{(1)}_{n-i+1}=\begin{pmatrix}
d_1 & e_2 & & & & & & \\
e_2 & d_2 & e_3 & & & & & \\
 & \ddots & \ddots & \ddots & & & & \\
0 & & e_{i-1} & d_{i-1} & e_i & & & \\
 & \ddots & & 0 & y_i & x_i & 0 & \\
 & & & & z_{i+1} & \omega_{i+1} & r_{i+1} & 0 \\
 & & & & & \ddots & \ddots & \ddots \\
 & & & & & z_n & \omega_n & r_n
\end{pmatrix}
$$

$$
A^{(1)}_{n-i+2}=\begin{pmatrix}
d_1 & e_2 & & & & & \\
e_2 & d_2 & e_3 & & & & \\
0 & \ddots & \ddots & \ddots & & & \\
 & \ddots & e_{i-2} & d_{i-2} & e_{i-1} & & \\
 & & 0 & y_{i-1} & x_{i-1} & 0 & \\
 & & & z_i & \omega_i & r_i & \ddots \\
 & & & & \ddots & \ddots & 0 \\
 & & & & z_n & \omega_n & r_n
\end{pmatrix}
$$

选择 $P_i^{(k)}$ 中的 c_{i-1} 和 s_{i-1} 使 $A^{(1)}_{n-i+2}$ 中 $(i-1,i)$ 处的元素为零，即

$$c_{i-1}e_i-s_{i-1}x_i=0$$

同时满足等式

$$(c_{i-1})^2+(s_{i-1})^2=1$$

解上面二个非线性方程组，得

$$s_{i-1}=\frac{e_i}{\sqrt{e_i^2+x_i^2}} \quad 和 \quad c_{i-1}=\frac{x_i}{\sqrt{e_i^2+x_i^2}}$$

按此继续旋转，最后得到下三角矩阵

$$
7 = A_n = \begin{pmatrix} r_1 & & & & \\ \omega_2 & r_2 & & & \\ z_3 & \omega_3 & r_3 & & \\ & \ddots & \ddots & \ddots & \\ & & z_n & \omega_n & r_n \end{pmatrix}
$$

定义

$Q^{(1)} = P_n^T P_{n-1}^T \cdots P_2^T$

因此，

$A^{(2)} = L^{(1)} Q^{(1)} = P_2 P_3 \cdots P_n A P_n^T P_{n-1}^T \cdots P_2$

这样，$A^{(2)}$ 变为三对角阵，其非对角线元素之绝对值一般比$A^{(1)}$ 相应的元素小，这就完成了一次迭代。重复这个过程，有 $A^{(3)}$，$A^{(4)}$，…。

8.11.2 具有移位的 QL 算法

8.11.4 具有移位的QL算法

$A^{(k)} - a_k I = Q^{(k)} L^{(k)}$ （QL分解）

$A^{(k+1)} = L^{(k)} Q^{(k)} + a_k I$

其中 a_k 是加速收敛的移位。从 $k=1$ 开始重复上述过程若干次之后，能使$e_1^{(k+1)}$ 收敛到零，这样 $d_1^{(k+1)} + \sum_{i=1}^{k} a_i$ 便是一个特征值。随后降一阶继续计算，直至获得全部特征值。

8.11.5 移位 a_k 的选择方法

设已获得 $r-1$个特征值，要计算的迭代是在从r行开始的子矩阵中进行，则取 a_k为2×2矩阵

$$
\begin{pmatrix} d_r^{(k)} & e_r^{(k)} \\ e_r^{(k)} & d_{r+1}^{(k)} \end{pmatrix}
$$

的接近 $d_r^{(k)}$ 的那个特征值，计算公式是

$$a_k = d_r^{(k)} - e_r^{(k)}/(p \pm \sqrt{1+p^2})$$

其中

$$p = (d_{r+1}^{(k)} - d_r^{(k)})/2e_r^{(k)}$$

式中正负号取与 p 的正负号相同，$p=0$时取正号。

8.11.6 QL算法的计算步骤

用 a 表示累加移位，$a = \sum_{i=1}^{k} a_i$。

1° $k:=1$，$a:=0$

2° 检验：

如果$e_2^{(k)} \approx 0$，则 $\lambda = d_1^{(k)} + a$，并输出 λ，

$n:=n-1$

$d_1^{(k)} := d_2^{(k)}$

对于$j=2,\cdots,n$执行

$d_j^{(k)} := d_{j+1}^{(k)}$

$e_j^{(k)} := e_{j+1}^{(k)}$

如果$e_j^{(k)} \approx 0 (3 \leqslant j \leqslant n-1)$，则表示矩阵需要分裂降阶并输出$d_1^{(k)},\cdots,d_{j-1}^{(k)},e_2^{(k)}\cdots e_{j-1}^{(k)}$和 $d_j^{(k)},\cdots,d_n^{(k)},e_{j+1}^{(k)},\cdots,e_n^{(k)}$，$a$，然后停机。

如果 $e_n^{(k)} \approx 0$，则 $\lambda := d_n^{(k)} + a$，并输出 λ，

$n:=n-1$

3° 计算移位a_k：

$b = -(d_2^{(k)} + d_1^{(k)})$

$c = d_2^{(k)} d_1^{(k)} - (e_2^{(k)})^2$

$g = (b^2 - 4c)^{1/2}$

如果 $b>0$，则 $\mu_1 = -2c/(b+g)$，$\mu_2 = -(b+g)/2$ 否则，$\mu_1 = (g-b)/2$，$\mu_2 = 2c/(g-b)$。

如果 $n=2$，则 $\lambda_1 = \mu_1 + a$，$\lambda_2 = \mu_2 + a$，输出 λ_1 和 λ_2，停机；否则选择a_k使

$|\alpha_k - d_1^{(k)}| = \min(|\mu_1 - d_1^{(k)}|, |\mu_2 - d_1^{(k)}|)$

4° 累加移位 $a := a + \alpha_k$

5° 执行移位：

$d_j = d_j^{(k)} - a \quad (j = 1, 2, \cdots n)$

6° 计算$L^{(k)}$：

$x_n = d_n$

$y_n = e_n^{(k)}$

对于 $j = n, n-1, \cdots, 2$做

$r_j = (x_j^2 + (e_j^{(k)})^2)^{\frac{1}{2}}$

$c_{j-1} = x_j / r_j$

$s_{j-1} = e_j^{(k)} / r_j$

$\omega_j = c_{j-1} y_j + s_{j-1} d_{j-1}$

$x_{j-1} = c_{j-1} d_{j-1} - s_{j-1} y_j$

如果 $j \neq 2$，则

$z_j = s_{j-1} e_{j-1}^{(k)}, \quad y_{j-1} = c_{j-1} e_{j-1}^{(k)}$

7° 计算 $A^{(k+1)}$：

$z_1 = x_1, \quad d_n^{(k+1)} = s_{n-1}\omega_n + c_{n-1} r_n, \quad e_n^{(k+1)} = s_{n-1} r_{n-1}$

对于 $j = n-1, n-2, \cdots, 2$有

$d_j^{(k+1)} = s_{j-1}\omega_j + c_{j-1} r_j$

$e_j^{(k+1)} = s_{j-1} r_{j-1}$

$d_1^{(k+1)} = c_1 x_1$

$k := k+1$，重复执行2°—7°，直到求出 A 的全部特征值为止。

8.11.7 例 用具有移位的 QL 算法求矩阵

$$A = A_1^{(1)} = \begin{pmatrix} 4 & 2 & 0 \\ 2 & 3 & 1 \\ 0 & 1 & 2 \end{pmatrix} = \begin{pmatrix} d_1^{(1)} & e_2^{(1)} & \\ e_2^{(1)} & d_2^{(1)} & e_3^{(1)} \\ & e_3^{(1)} & d_3^{(1)} \end{pmatrix}$$

的特征值。

解 首先求下列子矩阵

$$\begin{pmatrix} 4 & 2 \\ 2 & 3 \end{pmatrix}$$

的特征值，容易求得 $\mu_1=5.561552$ 和 $\mu_2=1.438447$，其中接近于 $d_1^{(1)}=4$ 的是 μ_1，取 $\alpha_1=\mu_1$。

$$A_1^{(1)}-\alpha_1 I=\begin{pmatrix} -1.561552 & 2 & 0 \\ 2 & -2.561552 & 1 \\ 0 & 1 & -3.561552 \end{pmatrix}=\begin{pmatrix} d_1 & e_2^{(1)} & \\ e_2^{(1)} & d_2 & e_3^{(1)} \\ & y_3 & x_3 \end{pmatrix}$$

$$P_3=\begin{pmatrix} 1 & 0 & 0 \\ 0 & -0.9627697 & -0.270323 \\ 0 & +0.270323 & -0.9627697 \end{pmatrix}=\begin{pmatrix} 1 & 0 & 0 \\ 0 & c_2 & -s_2 \\ 0 & s_2 & c_2 \end{pmatrix}$$

$$A_2^{(1)}=P_3(A_1^{(1)}-\alpha_1 I)$$

$$=\begin{pmatrix} -1.561552 & 2 & 0 \\ -1.9255392 & 2.195860 & 0 \\ 0.5406461 & -1.655216 & 3.699276 \end{pmatrix}=\begin{pmatrix} d_1 & e_2^{(1)} & 0 \\ y_2 & x_2 & 0 \\ z_3 & \omega_3 & r_3 \end{pmatrix}$$

$$P_2=\begin{pmatrix} 0.7393092 & -0.6733661 & 0 \\ 0.6733661 & 0.7393092 & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

$$L=A_3^{(1)}=P_2A_2^{(1)}=\begin{pmatrix} 0.1421232 & 0 & 0 \\ -2.475064 & 2.970151 & 0 \\ 0.5406461 & -1.655216 & 3.699276 \end{pmatrix}$$

$$=\begin{bmatrix} x_1 & & \\ \omega_2 & r_2 & \\ z_3 & \omega_3 & r_3 \end{bmatrix}$$

$$A^{(2)}=LP_3^TP_2^T=\begin{bmatrix} 0.1050729 & 0.09570091 & 0 \\ 0.09570091 & -3.780732 & 0.8029006 \\ 0 & 0.8029006 & -4.008994 \end{bmatrix}$$

选 $\alpha_2=0.1074284, \alpha=\alpha_1+\alpha_2=5.668980$，迭代得

$$A^{(3)}=\begin{bmatrix} 9.885922\times10^{-5} & 2.583158\times10^{-6} & 0 \\ 2.583158\times10^{-6} & -3.597252 & 0.7013983 \\ 0 & 0.7013983 & -4.409784 \end{bmatrix}$$

因为 $e_2^{(3)}=2.583158\times10^{-6}$ 已足够小，所以 $\lambda\approx d_1^{(3)}+\alpha=9.885922\times10^{-5}+5.668980=5.669077$ 为第一个特征值λ_1，由此算得$\lambda_2=2.476024, \lambda_3=0.8548984$。三个特征值的真值是5.669079087，2.476023608和0.8548973088。

8.12

8.12 广义特征值问题

设 A, B 是 n 阶矩阵，I 是 n 阶单位阵。求数 λ 和非零向量 x 使得

$(A-\lambda I)x=0$

这就是标准的特征值问题(Standard Eigenvalue Problem)。

求数 λ 和非零向量 x 使得

$Ax=\lambda Bx$

或

$ABx=\lambda x$

这样的问题称为广义特征值问题（Generalized Eigenvalue Problem）。把数 λ 和非零向量 x 称为问题 $Ax=\lambda Bx$ 或问题

$ABx=\lambda x$ 的特征值和特征向量．当B 是单位矩阵 时，则广义特征值问题化为标准特征值问题．

8.12.1 特征值问题 $Ax=\lambda Bx$

若 A是对称阵，B 是对称正定阵，则形如

8.12.1 $Ax=\lambda Bx$

的特征值问题，一般可用如下方法化为标准的对称阵的特征值问题。由于 B 正定，B 的 Cholesky 分解形式是

8.12.2 $B=LL^T$

其中L 是实的非奇异下三角阵．这种分解十分稳定，则（8.12.1）等价于

8.12.3 $Cy=\lambda y$

其中$C=L^{-1}A\ (L^{-1})^T$

8.12.4 $y=L^Tx$

C 的特征值 λ 就是问题（8.12.1）的特征值，两个问题的特征向量之间的关系是（8.12.4）．

由于 C 是对称阵，可以用 Givens 平面旋转变换或Householder 正交相似变换（8.8.2）约化为对称三对角阵，然后利用 Sturm序列的性质，采用二分法（8.9)求解，或者用 QR 算法（8.10）或 QL 算法（8.11）有效地求解．

当用二分法时，由于 C 是对称阵，矩阵 $C-\lambda I$ 具有(8.9)中所述的 Sturm 序列的性质，故矩阵 $A-\lambda B$ 具有同样的性质．如果 A_r, B_r, C_r 分别表示 A, B 和 C 的r 阶的首主子矩阵，则对所有的 r，$\det(A_r-\lambda B_r)$和 $\det(C_r-\lambda I)$ 的符号是相同的．

令 L_r 是 L 的 r阶首主子矩阵，则由(8.12.2) 有

$$L_rL_r^T=B_r$$

$$L_r^{-1}A_r(L_r^{-1})^T=C_r$$

于是

$$\det(A_r-\lambda B_r)=\det(L_r(L_r^{-1}A_r(L_r^{-1})^T-\lambda I)L_r^T)$$
$$=\det(L_r(C_r-\lambda I)L_r^T)$$
$$=\det(L_r)^2\det(C_r-\lambda I)$$

由于L_r是非奇异的，所以$\det(A_r-\lambda B_r)$和$\det(C_r-\lambda I)$的符号相同，因此，$A-\lambda B$大于λ的特征值的数目等于序列

$\det(A_r-\lambda B_r)\qquad(r=0,1,2,\cdots n)$

中相邻两项符号相同的数目，其中$\det(A_0-\lambda B_0)=1$，用二分法可求得各个特征值。

$A-\lambda B$一般不是正定的，故在求$A_r-\lambda B_r$的行列式的过程中选主元需要进行行交换，因此，第$(r+1)$个主子阵的行列式由

$\det(A_{r+1}-\lambda B_{r+1})=(-1)^M d_{11}d_{22}\cdots d_{r+1,r+1}$

给出，其中M是完成这一步所要交换的总次数，d_{ii}（$i=1,2,\cdots,r+1$）是$A_{r+1}-\lambda B_{r+1}$的对角线元素。

8.12.2 特征值问题ABx＝λx

设A和B是对称阵，两者或两者之一是正定的，有广义特征值问题

8.12.5 $AB\boldsymbol{x}=\lambda\boldsymbol{x}$

求解这类问题，通常不需要求矩阵A和B的乘积AB，因为AB未必对称，可以用如下既省运算量又稳定的方法。

若B正定，则B的Cholesky分解形式是

$B=LL^T$

其中L是实的非奇异的下三角阵。于是

$ALL^T\boldsymbol{x}=\lambda\boldsymbol{x}$

令

8.12.6 $\boldsymbol{y}=L^T\boldsymbol{x}$

$D=L^TAL$

则

8.12.7 $Dy=\lambda y$

因此，广义特征值问题(8.12.5)变成具有对称矩阵D的特征值问题(8.11.7)，而(8.12.6)式表示这两个问题的特征向量之间的关系。

9

第9章　非线性方程组数值解法

9.1

9.1 引　言

用计算机求解非线性方程组的有效算法，对求解各种非线性力学问题、电路问题、经济平衡问题、非线性规划以及各种非线性偏微分方程离散化得到的方程组，都有重要的实际意义。

非线性方程组的一般表达式为

9.1.1
$$\begin{cases} f_1(x_1,\cdots,x_n)=0 \\ \cdots \\ f_n(x_1,\cdots x_n)=0 \end{cases}$$

其中$n>1$，为正整数，f_i（$i=1$，…，n）是定义在实n维空间R^n中开域D上的实函数。若用向量记号

$$F(x)=\begin{pmatrix} f_1(x) \\ \vdots \\ f_n(x) \end{pmatrix},\quad x=\begin{pmatrix} x_1 \\ \vdots \\ x_n \end{pmatrix},\quad 0=\begin{pmatrix} 0 \\ \vdots \\ 0 \end{pmatrix}$$

则方程（9.1.1）可写成

9.1.2　$F(x)=0$

这里F表示定义在D上的非线性映射(Mapping)，记为$F:D\subset R^n\to R^n$。若存在$x^*\in D$使$F(x^*)=0$，则称x^*为方程（9.1.2）的解。当$n=1$时，就是方程求根问题(见第5章)。当方程（9.1.1）中所有f_i都是线性函数时，即为线性方程组；如果f_i中至少有一个是非线性函数，则称F为非线性方程

组。当$n>1$，F为非线性时，方程（9.1.1)的求解问题无论在理论或实际求解方法上，均比上述两种情形困难得多，因为非线性方程组解的情况复杂，它可能无解，也可能有任意多个解。

9.1.3 例 $f_1(x_1, x_2)=x_1^2-x_2+a=0$

$f_2(x_1, x_2)=-x_1+x_2^2+a=0$

这是$n=2$的例题，其中实数a在-1与1之间变化，在R^2中每个方程代表平面上一条曲线，求方程组的解就是求两条曲线交点，若取$a=1, \frac{1}{4}, 0, -1$，则四种情况的解见图(9.1.4)。

a) $a=1$，无解；

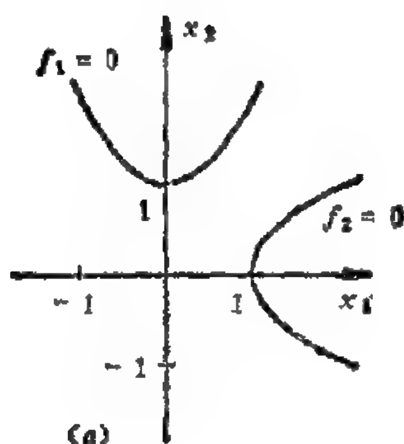
b) $a=\frac{1}{4}$，有唯一解$x_1=x_2=\frac{1}{2}$；

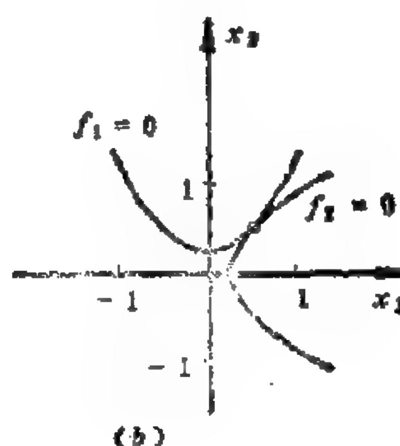
c) $a=0$，有2个解$x_1=x_2=0$；$x_1=x_2=1$；

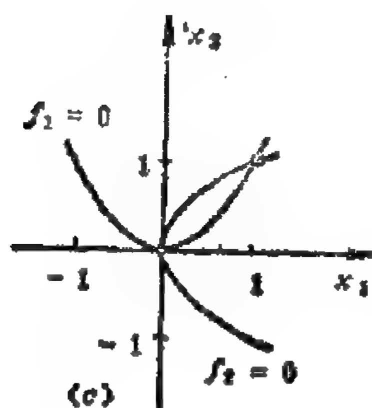
d) $a=-1$，有4个解，$x_1=-1, x_2=0$；$x_1=0$，$x_2=-1$；

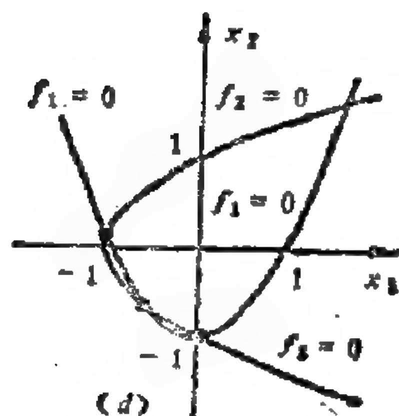
$x_1=x_2=\frac{1}{2}(1\pm\sqrt{5})$。

9.1.4 图









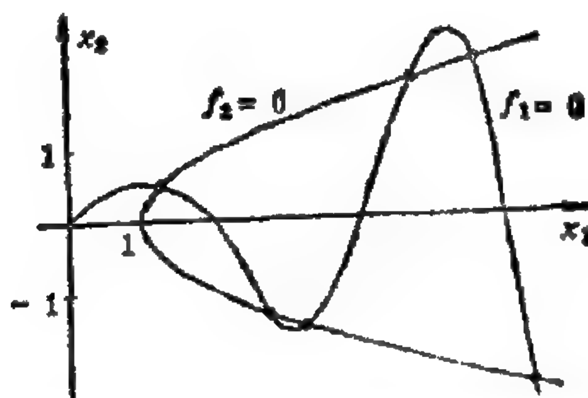
9.1.5 例 方程组

$$f_1(x_1,x_2)=\frac{1}{2}x_1\left[\sin\left(\frac{1}{2}\pi x_1\right)\right]-x_2=0$$

$$f_2(x_1,x_2)=x_2^2-x_1+1=0$$

有多个解，见图（9.1.6)。

9.1.6 图



为求解非线性方程组（9.1.2)，首先要知道解的情况，即有无解，有多少解；只要方程确实有解，就要研究描述和分析、逼近方程解的方法，主要是在计算机上使用较多的各种迭代算法和近十几年新发展的各种数值解法。

9.2 迭代法

9.2.1 迭代法的基本概念

为了研究迭代法，可把方程（9.1.2）改写成

9.2.1　$x=G(x)$

这里$G:D\subset \boldsymbol{R}^n\to \boldsymbol{R}^n$。例如$G(x)=x+BF(x)$，其中$B$为$n$阶非奇异矩阵，记$B\in \boldsymbol{R}^{n\times n}$。方程（9.1.2）的解$x^*$，也是方程（9.2.1）的解，即$x^*=G(x^*)$，$x^*$称为$G$的**不动点**(Fixed Point)。因此，求方程(9.2.1)的不动点等价于求方程(9.1.2)的解.求方程解的迭代法的过程，就是从$p\geqslant 1$个初始向量x^0，$\cdots,x^{p-1}$出发，通过某一迭代程序生成一个序列$\{x^k\}$的过程．例如，可构造迭代程序

9.2.2　$x^{k+1}=G(x^k)\quad (k=0,1,\cdots)$

它由x^0出发生成一个序列$\{x^k\}$，此时$p=1$，故称迭代法(9.2.2)为**单步法**(One-Step Methods)。当$p>1$时，迭代程序为

9.2.3　$x^{k+1}=G(x^k,\cdots,x^{k-p+1})\quad (k=p-1,p,\cdots)$

称为**多步法**(Multistep Methods)。应用迭代法有三个基本问题：

首先，是适定性问题．迭代法生成的序列$\{x^k\}$均在求解域D内，即$\{x^k\}\subset D$。例如，对迭代法（9.2.2）就要求每步x^k均能计算出结果，且所有$x^k\in D$。

其次，是收敛性问题。就是要求序列$\{x^k\}$收敛到方程（9.1.2）的解，即$\lim\limits_{k\to\infty} x^k=x^*$。由于对初始近似$x^0$的不同限制，迭代法收敛性有三种不同的概念。

1°　局部收敛性。

9.2.4　**定义**　假定$F:D\subset \boldsymbol{R}^n\to \boldsymbol{R}^n$，$x^*\in D$是方程（9.1.2）的一个解．若存在$x^*$的一个邻域$S\subset D$，使$\forall x^0\in S$，迭代序列

$\{x^k\}\subset S$，且收敛于x^*，则称迭代序列$\{x^k\}$具有**局部收敛性**(Local Convergence)，点 x^* 称为迭代序列 $\{x^k\}$ 的**吸引点**(Point of Attraction)。

2° 半局部收敛性．它不假定方程（9.1.2）解x^*存在，只在初始近似x^0满足一定条件下证明迭代序列$\{x^k\}$收敛于解x^*，这种收敛性定理本身包含了证明解的存在唯一性。实际上这时x^0仍在x^*附近，即$\| x^0-x^* \|$较小，因此，半局部收敛性定理对x^0的限制仍然较大。

3° 大范围收敛性．它对求解域D中任意的初始近似x^0，均能使序列$\{x^k\}$收敛到解x^*，是最完善的，也是最好的结果，但很多迭代法往往只具有局部或半局部收敛性。

第三，是效率问题．在用计算机求方程解的全过程中，机时是否节省，这是一个时间计算复杂性问题，对非线性方程组迭代法可用迭代序列$\{x^k\}$的收敛速度与每步迭代计算量大小来衡量。

9.2.5 定义 假定迭代序列$\{x^k\}$收敛于x^*，若存在实数$p\geqslant 1$，使当$k\geqslant k_0$，$x^k\neq x^*$时

$$0<\alpha_p=\lim_{k\to\infty}\frac{\| x^{k+1}-x^*\|}{\| x^k-x^*\|^p}<+\infty$$

成立，就说序列$\{x^k\}$是**p阶收敛**(Order p Convergence)，α_p称为**收敛因子**(Convergence Factors)。特别地，$p=1$称为**线性收敛**(Linear Convergence)，$p>1$称为**超线性收敛**(Superlinear Convergence)，$p=2$称为**平方收敛**(Quadratic Convergence)。

根据定义，一个迭代序列$\{x^k\}$的收敛阶越高说明它收敛越快，但效率高低还与每步迭代的计算量有关。

9.2.6 定义 假定迭代序列$\{x^k\}$的收敛阶为$p\geqslant 1$，每步迭代的计算量为w，则迭代序列的**效率**(Efficiency)e定义为

9.2.7 $e = \frac{\ln p}{w}$

对线性收敛($p=1$)的迭代序列，效率e可定义为$e=-\frac{\ln a_1}{w}$，这里$0<a_1<1$为收敛因子。

9.2.2 压缩映射原理与迭代法收敛性

对于方程(9.2.1)的解，其存在唯一性及迭代序列收敛性有以下定义与定理。

9.2.8 **定义** 假定$G: D\subset \boldsymbol{R}^n \to \boldsymbol{R}^n$，若存在常数$0<a<1$，使$\forall x, y\in D_0\subset D$有

9.2.9 $\| G(x)-G(y) \| \leqslant a \| x-y \|$

成立，则称G在D_0上为**压缩映射**(Contraction Mapping)。

9.2.10 **定理/压缩映射原理** 设$G: D\subset \boldsymbol{R}^n \to \boldsymbol{R}^n$为闭集$D_0\subset D$上的压缩映射，且$GD_0\subset D_0$，则$G$在$D_0$中存在唯一的不动点$x^*$。而且，从任何$x^0\in D_0$出发，由迭代法(9.2.2)生成的序列$\{x^k\}$均收敛于$x^*$。并有误差估计

9.2.11 $\| x^k - x^* \| \leqslant \frac{a^k}{1-a} \| x^1 - x^0 \|$

定理(9.2.10)给出了方程(9.2.1)解存在唯一性的条件，只有在这个条件下，迭代法(9.2.2)是大范围收敛的。但在实际计算时，构造的迭代法往往不能满足定理(9.2.10)条件，而只在解x^*附近满足定理条件，即条件较弱的局部收敛定理。

9.2.12 **定理** 设x^*是方程(9.2.1)的解，$G: D\subset \boldsymbol{R}^n \to \boldsymbol{R}^n$，若存在开球$S=\{x \mid \| x-x^* \| <\delta, \delta>0\}=S(x^*, \delta)\subset D$和常数$0<a<1$，使$\forall x\in S$，有

9.2.13 $\| G(x)-G(x^*) \| \leqslant a \| x-x^* \|$

则$\forall x^0\in S$，由迭代法(9.2.2)生成的序列$\{x^k\}$仍在S中，且

线性收敛于x^*.

这定理给出的收敛条件（9.2.13）往往不易实现，在研究迭代法收敛性时，普遍用到映射的导数概念.

9.2.14 定义 假定$F:D\subset \boldsymbol{R}^n\to \boldsymbol{R}^m$，如果对任何固定的$h\in R^n$，存在矩阵$A\in \boldsymbol{R}^{n\times m}$，使

$$\lim_{t\to 0}\frac{1}{t}\|F(x+th)-F(x)-t(Ah)\|=0$$

成立，则称映射F在x处为Gateaux（伽多）-可导，简称G-可导，并称A为F的**G-导数**(G-derivative)，记$A=F'(x)$。如果有

$$\lim_{\|h\|\to 0}\|F(x+h)-F(x)-Ah\|=0$$

则称映射F在x处为Frechet(弗瑞歇)-可导，简称F-可导.称A为映射F的**F-导数**（F-derivative）.

根据定义，当F在x处F-可导，则一定也是G-可导，且其导数均为$F'(x)$。而当F在x处G-可导，则在x处F的各分量f_1，…，f_m偏导数$\dfrac{\partial f_i}{\partial x_j}$（$i=1$，…，$m$，$j=1$，…，$n$）均存在，且

$$F'(x)=\begin{pmatrix}\dfrac{\partial f_1}{\partial x_1} & \cdots & \dfrac{\partial f_1}{\partial x_n}\\ \vdots & \cdots & \vdots \\ \dfrac{\partial f_m}{\partial x_1} & \cdots & \dfrac{\partial f_m}{\partial x_n}\end{pmatrix} \tag{9.2.15}$$

称为F的**Jacobi(雅可比)矩阵**.

利用导数定义可由以下定理条件代替迭代序列(9.2.2)的收敛条件(9.2.13).

9.2.16 定理 设映射$G:D\subset \boldsymbol{R}^n\to \boldsymbol{R}^n$有一不动点$x^*\in \mathrm{int}(D)$，且$G$在$x^*$处F-可导，$G'(x^*)$的谱半径

9.2.17 $\rho(G'(x^*))=\sigma<1$

则存在开球$S=S(x^*,\delta)=\{x\in R^n|\ \|x-x^*\|\leqslant\delta\}\subset D$,对任意初始近似$x^0\in S$，$x^*$是迭代序列(9.2.2)的吸引点．

这个定理给出的条件可做为判断具体迭代序列收敛的依据。

9.3

9.3 Newton法及其变形

9.3.1 Newton法

求解方程（9.1.1）的Newton 法是$n=1$的Newton 法(9.4.3)的推广，假定x^*是方程的解，它的k次近似$x^k\in D$，利用多元Taylor展开，可得

$$0=F(x^*)=F(x^k)+F'(x^k)(x^*-x^k)+R(x^*-x^k)$$

其中 $\lim\limits_{\|\Delta x\|\to0}\dfrac{\|R(\Delta x)\|}{\|\Delta x\|}=0$

因此，当x^k靠近x^*时，可略去余项$R(x^*-x^k)$，从而用线性方程组

9.3.1 $F'(x^k)\Delta x=-F(x^k)$

的解Δx近似差x^*-x^k，记$\Delta x=x^{k+1}-x^k$。于是可得

$$x^{k+1}=x^k+\Delta x=x^k-F'(x^k)^{-1}F(x^k)$$

作为x^*的新近似，可得到迭代程序：

9.3.2 $x^{k+1}=x^k-F'(x^k)^{-1}F(x^k)\quad(k=0,1,\cdots)$

此即为解非线性方程组的Newton法，这里$F'(x)$是F的Jacobi矩阵(9.2.15).公式（9.3.2）只是一种形式记号，实际计算时求逆就是解方程（9.3.1），故可将（9.3.2）改为下列形式：

9.3.3 $\begin{cases}x^{k+1}=x^k+\Delta x^k\\ F'(x^k)\Delta x^k=-F(x^k)\quad(k=0,1,\cdots)\end{cases}$

Newton法由x^k计算x^{k+1}的步骤是：

1° 计算$F(x^k)$及$F'(x^k)$；

2° 解线性方程组$F'(x^k)\Delta x^k=-F(x^k)$，求得$\Delta x^k$；

3° 令$x^{k+1}=x^k+\Delta x^k$。

9.3.4 例 考虑方程组

$$\begin{cases} f_1(x_1,x_2)=4x_1^2+x_2^2-4=0 \\ f_2(x_1,x_2)=x_1+x_2-\sin(x_1-x_2)=0 \end{cases}$$

在$(x_1^0,\ x_2^0)=(1,\ 0)$附近的解。

解 求F的Jacobi矩阵

$$F'(x)=\begin{bmatrix} \dfrac{\partial f_1}{\partial x_1} & \dfrac{\partial f_1}{\partial x_2} \\ \dfrac{\partial f_2}{\partial x_1} & \dfrac{\partial f_2}{\partial x_2} \end{bmatrix}=\begin{bmatrix} 8x_1 & 2x_2 \\ 1-\cos(x_1-x_2) & 1+\cos(x_1-x_2) \end{bmatrix}$$

从$(x_1^0,\ x_2^0)=(1,\ 0)$ 出发，计算

$$F'(x^0)=\begin{bmatrix} 8 & 0 \\ 0.4597 & 1.5403 \end{bmatrix},\qquad F(x^0)=\begin{bmatrix} 0 \\ 0.159 \end{bmatrix}$$

由公式 (9.3.3) 算出$x_1^1=1.0$，$x_2^1=-0.10292$，再由$(x_1^1,\ x_2^1)$出发，由公式 (9.3.3) 逐步迭代，表 (9.3.5)给出了计算结果。

9.3.5 表 Newton 法计算例题

k	x_1^k	x_2^k	$f_1(x_1^k,x_2^k)$	$f_2(x_1^k,x_2^k)$
0	1.0	0.0	0.0	1.59E-1
1	1.0	−.1029207	1.06E-2	4.55E-3
2	0.998609	−.1055307	1.46E-5	6.63E-7
3	0.998607	−.1055305	2.20E-11	−1.03E-11

9.3.2 Newton法的收敛性

有关Newton法的收敛定理很多，但最重要最著名的是Канторович（康托洛维奇）定理

9.3.6 定理/Канторович 假定$F: D\subset R^n\to R^n$在凸集$D_0\subset D$上连续可微，且满足条件：

1° $\| F'(x)-F'(y)\| \leqslant \gamma \| x-y\|, \forall x, y\in D_0$

2° $\| F'(x^0)^{-1}F(x^0)\| \leqslant \eta$, $x^0\in D_0$

3° $\| F'(x^0)^{-1}\| \leqslant \beta$

4° 令 $h=\beta\gamma\eta\leqslant\frac{1}{2}$, $t^*=\frac{\eta}{h}[1-\sqrt{1-2h}]$

$t^{**}=\frac{\eta}{h}[1+\sqrt{1-2h}]$

$\overline{S}(x^0, t^*)=\{x\in R^n | \| x-x^0\| \leqslant t^*\}\subset D$

则Newton法（9.3.2）生成序列$\{x^k\}\subset \overline{S}(x^0, t^*)$，且收敛到方程$F(x)=0$的一个解$x^*$，在域$S(x^0, t^{**})\cap D_0$内解$x^*$是唯一的，其误差估计

9.3.7 $$\| x^*-x^k\| \leqslant \frac{\eta}{h}\frac{(1-\sqrt{1-2h})^{2^k}}{2^k} \quad (k=0,1,\cdots)$$

这是一个半局部收敛定理，它既证明了Newton迭代序列$\{x^k\}$收敛到方程的解，也证明了方程解在x^0的一个邻域内存在唯一．根据这一定理还可证明Newton法是平方收敛的．

Newton法具有收敛快的优点，在计算时每一步计算只与前一步有关，误差不传播，是自校正的，因此，在理论和实际应用上都是一种重要方法．但是Newton法对初始近似x^0限制较大，只有x^0与解x^*靠近才能保证收敛．另外，每步要算n^2个分量偏导数值和n个分量函数值，还要解一个线性方程组，故工作量较大．针对这些缺点Newton法有不少改进的算法．

9.3.3 Newton-Steffensen方法

在实际使用Newton 法时，为了避免计算 $F'(x)$，通常可用差商代替偏导数，即用矩阵$J(x, h)$代替$F'(x)$.

9.3.8
$$J(x,h)=\begin{pmatrix} \frac{1}{h_1}[f_1(x+h_1e_1)-f_1(x)],\cdots, & \frac{1}{h_n}[f_1(x+h_ne_n)-f_1(x)] \\ \cdots \qquad\qquad \cdots & \cdots \\ \frac{1}{h_1}[f_n(x+h_1e_1)-f_n(x)],\cdots, & \frac{1}{h_n}[f_n(x+h_ne_n)-f_n(x)] \end{pmatrix}$$

这里$h=(h_1, \cdots, h_n)^T$,e_i为第i个坐标向量.如果$J(x, h)^{-1}$存在，用$J(x^k, h^k)^{-1}$代替（9.3.2）的$F'(x^k)^{-1}$，则得**离散Newton法**：

9.3.9 $x^{k+1}=x^k-J(x^k,h^k)^{-1}F(x^k)\quad(k=0,1,\cdots)$

其中h^k为预先给定的向量序列。若取

$h^k=(f_1(x^k),\cdots,f_n(x^k))^T=F(x^k)$

这时$J(x^k, h^k)=J(x^k, F(x^k))$，将它代入（9.3.9）则得

9.3.10 $x^{k+1}=x^k-J(x^k,F(x^k))^{-1}F(x^k)\quad(k=0,1,\cdots)$

此即Newton-Steffensen（**牛顿-斯蒂芬森**）方法.这个算法同样具有平方敛速，每步计算$(n+1)$个$F(x)$函数值，如果计算工作量W以每步计算函数值的个数衡量，则$W=n+1$，而收敛阶$p=2$，于是由（9.2.7）可知此算法效率

9.3.11 $e_N=\dfrac{\ln 2}{n+1}$

在Newton法中，如果把计算$F'(x^k)$与计算n个$F(x^k)$等同，则其效率也为e_N。程序（9.3.10）仅把计算偏导数变成

计算函数值，以便于在计算机上使用，其余的效果与Newton法相同。

9.3.4 Newton–Шаманский 方法

为了减少Newton法的计算量，通常可用简化Newton法，即

9.3.12 $x^{k+1}=x^k-F'(x^0)^{-1}F(x^k) \quad (k=0,1,\cdots)$

它除开始计算$F'(x^0)^{-1}$外，以后每步只计算一个函数值，计算量省，但这个程序只有线性敛速，收敛慢，故算法(9.3.12)效率仍很低。Шаманский（沙曼斯基）把m步简化Newton法组成一步Newton法，得到下列程序：

9.3.13 $$\begin{cases} x^{k,0}=x^0 \\ x^{k,i}=x^{k,i-1}-F'(x^k)^{-1}F(x^{k,i-1}) \\ \qquad (i=1,\cdots,m, \quad k=0,1,\cdots) \\ x^{k+1}=x^{k,m} \end{cases}$$

称为Newton–Шаманский方法。这个程序由x^k计算到x^{k+1}只计算一次$F'(x^k)$，相当于计算n个F值，求一次$F'(x^k)^{-1}$，和m个F的函数值，总共计算函数值的次数为$n+m$，即$w=n+m$。程序（9.3.13）生成的序列具有$m+1$阶 敛速，即 $p=m+1$。这个算法的效率

9.3.14 $$e_{N_m}=\frac{\ln(m+1)}{n+m}$$

当$m=1$时即是Newton法。$e_{N_1}=e_N$，由此又有

9.3.15 $$\frac{e_{N_m}}{e_N}=\frac{n+1}{n+m}\cdot\frac{\ln(m+1)}{\ln 2} \qquad m\leqslant n+1$$

当$m\geqslant 2$时显然有$\frac{e_{N_m}}{e_N}>1$，这说明算法（9.3.13）的效率高

于Newton 法。对给出的n，如果可以选最优的m，使(9.3.15) 取最大值，则有相应的结果（表9.3.16）.

9.3.16 表 Шаманский法与Newton法的效率比

n	2	3	5	10	50	100	1000
m	3	3	6	7	22	37	225
$\frac{e_{N_m}}{e_{N_1}}$	1.20	1.33	1.65	1.94	3.20	3.87	6.39

迭代法 (9.3.13) 比Newton法效率高，又包含了 Newton法，因此，它比Newton 法更有效，是一个可供实际使用的算法。计算时，$F'(x^k)$也可用$J(x^k, F(x^k))$代替。

9.3.5 Newton 下山法

为了扩大Newton法的收敛范围，通常可构造 Newton 下山程序

9.3.17 $x^{k+1}=x^k-\omega_k F'(x^k)^{-1}F(x^k) \quad (k=0,1,\cdots)$

其中ω_k称为下山参数，$0<\omega_k\leqslant 1$，可选择ω_k使

9.3.18 $\| F(x^{k+1}) \| < \| F(x^k) \|$

成立。由于Newton法 (9.3.2) 没有保证下山条件(9.3.18)成立，所以引入参数ω_k以后，就可使(9.3.18)的条件成立，从而使迭代序列收敛。

9.3.19 **定理** 设$F: D\subset R^n \to R^n$连续可微，$\| F(x^0) \| \leqslant \eta$，在域$D_0=\{x \mid \| x-x^0 \| \leqslant \rho\beta\eta\}\subset D$上$F'(x)$可逆，且

$\| F'(x)^{-1} \| \leqslant \beta$，此外

$\| F'(x)-F'(y) \| \leqslant \gamma \| x-y \|, \forall x,y\in D_0$

成立。令$\alpha=\gamma\beta^2\eta$，则当$\rho\geqslant \frac{2\alpha}{\varepsilon^2}>1$时方程 (9.1.1) 在$D_0$中有解$x^*$存在，且在$0<\omega_k\leqslant 1$及$0<\varepsilon\leqslant w_k\alpha\leqslant 2-\varepsilon$时，由

(9.3.17) 生成的序列$\{x^k\}$至少线性收敛于x^*.

从定理可知，当给定$x^0\in D$后，每步迭代总可找到满足定理要求的参数$\omega_k\leqslant 1$，使迭代法（9.3.17）收敛。在这个意义下Newton下山法是大范围收敛的。但在实际计算时，要选择满足定理条件的ω_k仍较困难，因此，可以先令$\omega_k=1$，用逐次减半方法确定ω_k，只要检验条件（9.3.18）成立即可。这样做当然增加了计算量，但却减少了对初始近似x^0的限制。实际计算时还可用$J(x^k, F(x^k))^{-1}$代替$F'(x^k)^{-1}$，从而使（9.3.17）改为

9.3.20 $$x^{k+1}=x^k-\omega_k J(x^k,F(x^k))^{-1}F(x^k)\quad (k=0,1,\cdots)$$

9.4

9.4 Brown方法与Brent方法

9.4.1 Brown方法

在Newton法（9.3.2）中，把非线性映射F作为一个整体逐次线性化，然后求解Newton方程组（9.3.1）得到Δx^k。这种方法便于进行理论分析，又能简化迭代格式表示，但未充分利用函数F的具体结构。当各分量函数$f_i(i=1, \cdots, n)$非线性程度分布不均时，把所有分量做为一个整体考虑显然对计算效率的提高是不利的，为此，有必要按分量方程$f_i(x)=0(i=1, \cdots, n)$逐个加以考虑。Brown（布朗）正是基于这种考虑，于1966年首先提出了对Newton法的改进，Brown算法的基本思想是对F各个分量f_i逐个线性化，并用其中每一个线性方程消去余下那些非线性方程的一个变量，得到一个上三角形的线性方程组，再逐个回代求得新近似x^{k+1}的各分量。这样做减少了计算F函数值的个数，又具有平方敛速，从而提高了效率。Brown方法很大程度依赖于F和x的分量顺序，为着方便，叙述是以原分量次序顺序进行，但使用时则应按一定原则排列次序。设方程（9.1.2）的k次近

似解为$x^k=(x_1^k, \cdots, x_n^k)$，由$x^k$计算$x^{k+1}$的步骤如下：

第1步，对方程$Fx=0$的第一个方程$f_1(x)=0$用线性方程

$$9.4.1 \quad f_1(x^k)+\sum_{j=1}^{n}\Delta_{1j}^k(x_j-x_j^k)=0$$

近似代替，其中

$$\Delta_{1j}^k=\frac{1}{h_j^k}[f_1(x^k+h_j^k e_j)-f_1(x^k)]\approx\frac{\partial f_1(x^k)}{\partial x_j}$$

$$(j=1,\cdots,n)$$

这里实际上是用$f_1(x)$在点x^k的一阶Taylor近似逼近$f_1(x)$，但为避免计算偏导数$\frac{\partial f_1(x^k)}{\partial x_j}$而用差商代替。假定$\Delta_{11}^k\neq 0$，且在$\Delta_{1j}^k$（$j=1, \cdots, n$）中绝对值最大，则由（9.4.1）可求得

$$9.4.2 \quad x_1=x_1^k-(\Delta_{11}^k)^{-1}[f_1(x^k)+\sum_{j=2}^{n}\Delta_{1j}^k(x_j-x_j^k)]$$

$$=L_1(x_2,\cdots,x_n)$$

在实际计算时，应选Δ_{1j}中绝对值最大者，求出x_j，即选主元。

第2步，将（9.4.2）的x_1代入方程（9.1.2）其余各方程，并记

$$g_2(x_2,\cdots,x_n)=f_2(L_1(x_2,\cdots,x_n),x_2,\cdots,x_n)$$

$$g_2^k=f_2(L_1^k,x_2^k,\cdots,x_n^k),\ L_1^k=L_1(x_2^k,\cdots,x_n^k)$$

用与第1步同样的方法，对$g_2(x_2, \cdots, x_n)$用x^k处一阶Taylor近似，并解出x_2，得到

$$x_2=x_2^k-(\Delta_{22}^k)^{-1}\left[g_2^k+\sum_{j=3}^{n}\Delta_{2j}^k(x_j-x_j^k)\right]=L_2(x_3,\cdots,x_n)$$

其中

$$\Delta_{2j}^k=\frac{1}{h_j^k}[g_2(x_2^k,\cdots,x_j^k+h_j^k e_j,\cdots,x_n^k)-g_2^k]$$

$$(j=2,\cdots,n)$$

第i步，将$x_1=L_1(x_2, \cdots,x_n),\cdots,x_{i-1}=L_{i-1}(x_i,\cdots,x_n)$

代入其余$n-i+1$个方程，记

$g_i(x_i, \cdots, x_n)=f_i(L_1, \cdots, L_{i-1}, x_i, \cdots, x_n)$

其中L_i是由$i-1$列三角线性方程组回代得到，所有L_i都要用$x_i, \cdots, x_n$表示，用与前面步骤相同的过程可求得

9.4.3 $$x_i=x_i^k-(\Delta_{i\,i}^k)^{-1}\left[g_i^k+\sum_{j=i+1}^{n}\Delta_{i\,j}^k(x_j-x_j^k)\right]$$

$$=L_i(x_{i+1},\cdots,x_n)$$

其中

9.4.4 $$\Delta_{i\,j}^k=\frac{1}{h_j^k}[g_i(x_i^k,\cdots x_{j-1}^k,x_j^k+h_j^k e_j,x_{j+1}^k,\cdots,x_n^k)-g_i^k]$$

$$(j=i,\cdots,n)$$

这样每步消去一个变量。

第n步，由公式（9.4.3），当i从1算到n时有

$g_n(x_n)=f_n(L_1, \cdots, L_{n-1}, x_n)$

其中L_i（$i=1, \cdots, n-1$），都是x_n的函数，由此求出

$x_n=x_n^k-(\Delta_{n\,n}^k)^{-1}g_n^k=L_n$

把它作为x^*的第n个分量x_n^*的第$k+1$次近似，记作x_n^{k+1}，即

$x_n^{k+1}=x_n^k-(\Delta_{n\,n}^k)^{-1}g_n^k$

其中$\Delta_{n\,n}^k$是$g_n(x_n)$在点x_n^k处的导数近似。将它代入（9.4.3）这就是“回代过程”，可逐步得到

9.4.5 $$x_i^{k+1}=L_i(x_{i+1}^{k+1},\cdots,x_n^{k+1}) \quad (i=n-1,\cdots,2,1)$$

按以上步骤从x^0开始反复迭代，直到x^m满足精度要求为止（Brown算法程序可见附录）。考虑到步长与函数值、变量值及机器字长等的关系，对第i个分量函数g_i，h_j^k按如下规定选取：

$h_j^k=\max\{a_{i\,j}^k,5\times10^{-\beta+2}\}$（$j=1,2,\cdots,n$）

而

$a_{i\,j}^k=\min\{\max(|f_1^{(k)}|,|g_2^k|,\cdots,|g_i^k|),0.01x|x_j^k|\}$

其中β为机器的有效数位（十进制）。

根据以上算法可知，Brown算法每迭代一步所需计算函数值个数是：$i=1$时要算f_1的$n+1$个值，$i=2$时要算f_2的n个值，依此类推，由x^k到x^{k+1}迭代一步需要计算

$$\sum_{i=2}^{n+1} i=\frac{n^2+3n}{2}$$

个分量函数值，相当于$\frac{n+3}{2}$个F值，比Newton法节省将近一半计算量。Brown算法生成的序列$\{x^k\}$是平方收敛的，它的效率

$$e_{B_1}=\frac{2\ln 2}{n+3}$$

因为Brown方法每次只对一个分量方程$f_i=0$进行运算，所以就存在一个最优次序问题，使用时应先把线性方程排在前面，然后按非线性程度由低次到高次排列，这也是Brown算法的一个优点。

9.4.6　**例**　设非线性方程组为

$$\begin{cases} x_1^2+x_2^2-x_3-2=0 \\ x_1+5x_2+1=0 \\ x_1x_3-2x_1+1=0 \end{cases}$$

用Brown方法求$x^0=(-2,\ 0,\ 2)$附近的根。

解　先将给定方程组按线性情况排列，令

$$\begin{cases} f_1(x)=x_1+5x_2+1=0 \\ f_2(x)=x_1x_3-2x_1+1=0 \\ f_3(x)=x_1^2+x_2^2-x_3-2=0 \end{cases}$$

因f_1已是线性，按主元消去x_2，则得

$$x_2=-\frac{1}{5}(x_1+1)=L_1(x_1,x_3)$$

代入f_2，f_3，对f_2用线性函数近似，即用

$$l_2(x)=f_2^k+(x_3^k-2)(x_1-x_1^k)+x_1^k(x_3-x_3^k)=0$$

近似$f_2(x)=0$，解出

$$x_3=x_3^k-\frac{1}{x_1^k}[(x_3^k-2)x_1+1]=L_2(x_1)$$

将它代入f_3，得

$$g_3(x_1)=x_1^2+\frac{1}{25}(x_1+1)^2$$

$$-\left\{x_3^k-\frac{1}{x_1^k}[(x_3^k-2)x_1+1]\right\}-2$$

对$g_3(x_1)=0$用一步Newton法，得

9.4.7 $$x_1^{k+1}=x_1^k-\frac{(x_1^k)^2+\frac{1}{25}(x_1^k+1)^2-\left\{x_3^k-\frac{1}{x_1^k}[(x_3^k-2)x_1^k+1]\right\}-2}{\frac{52}{25}x_1^k+\frac{2}{25}+\frac{x_3^k-2}{x_1^k}}$$

$(k=0,1,\cdots)$

将得到的x_1^{k+1}进行回代，得到

9.4.8 $$\begin{cases} x_2^{k+1}=-\frac{1}{5}(x_1^{k+1}+1) \\ x_3^{k+1}=x_3^k-\frac{1}{x_1^k}[(x_3^k-2)x_1^{k+1}+1] \quad (k=0,1,\cdots) \end{cases}$$

用（9.4.7）和（9.4.8）的迭代过程，从x^0出发可得到下列计算结果

k	x_1^k	x_2^k	x_3^k
0	−2.000000	0.000000	2.000000
1	−2.112747	0.222549	2.500000
2	−2.103978	0.220796	2.475393
3	−2.103937	0.220787	2.475299
4	−2.103937	0.220787	2.475299

在例（9.4.6）中，Δ_{ij}均用偏导数$\frac{\partial f_i}{\partial x_j}$，而不用差商近似，但在计算机上为避免算偏导数，通常采用差商近似Δ_{ij}。

9.4.2 Brent方法

Brent（布兰特）于1973年在Brown方法基础上使用Шаманский技巧，即在由x^k计算x^{k+1}时，用了m步Brown方法，且每步用的Δ^k_{ij}不变。这时，计算回代过程的公式可表示为：

$$
9.4.9\quad \begin{cases} x_i^{k,l+1}=x_i^{k,l}-(\Delta_{ii}^k)^{-1}\Big[g_i(x^{k,l}) \\ \qquad\qquad +\sum\limits_{j=i+1}^{n}\Delta_{ki}^{j}(x_k^{j,l+1}-x_i^{k,l})\Big] \\ x_n^{k,l+1}=x_n^{k,l}-(\Delta_{nn}^k)^{-1}g_n^{k,l} \end{cases}
$$

$$(i=n-1,\cdots,1,\quad l=0,1,\cdots,m-1)$$

其中$x^{k,0}=x^k$，$x^{k,m}=x^{k+1}$，Δ^k_{ij}由（9.4.4）表示。这就是Brent方法的基本思想。这时由x^k计算x^{k+1}的工作量比一步Brown法大，多算了（$m-1$）个函数值。故总工作量为$w=\frac{n+3}{2}+m-1=\frac{n+2m+1}{2}$次函数值。

然而，Brent方法在具体实现时并不采用Brown方法的消去与回代算法，而使用一种基于正交变换的方法，即假定k次迭代后，已有x^*的k次近似x^k及一个正交矩阵Q_k和一个步长$h_k>0$（初始x^0，$h_0>0$，，给定$Q_0=I$），则Brent方法生成x^{k+1}，h_{k+1}，及Q_{k+1}的计算步骤如下：

1° 令$Q_{k,1}=Q_k$，$y^{k,1,0}=x^k$；

2° 对$j=1,\cdots,n$做3°—5°的步骤；

3° 计算

$$a_{k,j}=\frac{1}{h_k}\begin{pmatrix}0\\ \vdots\\ 0\\ f_i(y^{k,j,0}+h_kQ_{k,j}e_i)-f_i(y^{k,j,0})\\ \vdots\\ f_j(y^{k,j,0}+h_kQ_{k,j}e_n)-f_j(y^{k,j,0})\end{pmatrix}$$

4° 令$\sigma_{k,j}=\pm\|a_{k,j}\|$，找一个正交矩阵$U_{k,j}$，形如

$$U_{k,j}=\left(\begin{array}{c:c}I_{j-1}&0\\ \hdashline 0&\widehat{U}_{k,j}\end{array}\right)\begin{matrix}\}j-1\\ \}n-j+1\end{matrix}$$

使$U_{k,j}^Ta_{k,j}=\sigma_{k,j}e_j$，这里$\widehat{U}_{k,j}$取为初等反射阵。于是$U_{k,j}$可表示为

$U_{k,j}=I-2w_jw_j^T$

其中$w_j\in R^n$，具有形式

$w_j^T=(0,\cdots,0,w_{jj},\cdots\cdots,w_{j,n})$

且$w_j^Tw_j=1$。

5° 计算

$Q_{k,j+1}=Q_{k,j}U_{k,j}$

及 $y^{k,j+1,0}=y^{k,j,0}-\left(\dfrac{f_j(y^{k,j,0})}{\sigma_{k,j}}\right)Q_{k,j+1}e_j$

6° 对$l=1,\cdots,m$，令$y^{k,1,l}=y^{k,n+1,l-1}$，对$j=1,\cdots,n$计算

$$y^{k,j+1,l}=y^{k,j,l}-\left(\frac{f_j(y^{k,j,l})}{\sigma_{k,j}}\right)Q_{k,n+1}e_j$$

7° 令$x^{k+1}=y^{k,n+1,m}$，计算Fx^{k+1}，若$\|Fx^{k+1}\|\leqslant\varepsilon_2$或$\|x^{k+1}-x^k\|\leqslant\varepsilon_1\|x^{k+1}\|$（其中$\varepsilon_1,\varepsilon_2$为给定精度要求），则

置$x^*=x^{k+1}$，打印x^*后结束；否则$k+1\to k$，$x^{k+1}\to x^k$，$Q_{k+1}=Q_{k,n+1}\to Q_k$

$$h_{k+1}=\begin{cases}-f_1(x^{k+1})/\sigma_{k,1}, & 当 f_1(x^{k+1})\neq 0\\ \varepsilon\|x^{k+1}\|, & 当 f_1(x^{k+1})=0\end{cases}$$

$h_{k+1}\to h_k$，转步骤1°。

9.4.10 定理 假定$\varepsilon>0$，$n>1$，$S=\{x\in R^n \mid \|x-x^*\|<3\varepsilon\}$，$F$：$S\to R^n$是F-可微，且$F(x^*)=0$。若$F'(x^*)$非奇，$h_0<\varepsilon$且$\|x^0-x^*\|<\varepsilon$，当$\varepsilon$充分小，由Brent方法生成的序列$\{x^k\}$收敛于$x^*$，且收敛阶至少为$m+1$。

根据（9.2.7），Brent方法的效率

9.4.11 $$e(B_m)=\frac{2\ln(m+1)}{n+2m+1}$$

当$m=1$时，

$$e(B_1)=e_{B_1}=\frac{2\ln 2}{n+3}$$

在（9.4.11）中当n给定时，可求出最优的m，记作$\bar{m}$，即$e(B_{\bar{m}})=\max\limits_m e(B_m)$，$\bar{m}$值见表（9.4.12）。

9.4.12 表 Brown 法与Brent 法的效率比

n	2	3	5	8	10	20	30	50	100	1000
$\bar{m}$	2	3	3	4	5	7	10	14	22	128
$e(B_1)/e(N_1)$	1.20	1.33	1.50	1.64	1.69	1.83	1.88	1.92	1.96	2.00
$e(B_{\bar{m}})/e(N_1)$	1.36	1.60	2.00	2.46	2.71	3.60	4.21	5.04	6.30	11.17

Brent方法取最优m值时效率最高，因此，目前不少计算机的数学库中都有用Brent方法编制的软件。

9.5 Newton松弛型迭代法

在Newton法中，每步迭代都要解一个线性方程组

9.5.1 $F'(x^k)\Delta x^k = -F(x^k)$ $(k=0,1,\cdots)$

此即为Newton方程组。对有的问题（如大型，稀疏方程）用迭代法求解（9.5.1）比直接法更方便。如用迭代法解，则在Newton法每步中又生成一个解线性方程组的迭代序列$\{x^{k,j}, j=0,1,\cdots\}$，这就形成一个双层迭代，其中$k=0,1,\cdots$表示Newton迭代，对每个$k$又有$j=0,1,\cdots$表示的线性方程组迭代做辅助迭代。从理论上说，一切求解线性方程组的迭代法都可用于求方程（9.5.1）的解，其中，用SOR迭代求解Newton方程组被称为Newton-SOR迭代，简记N-SOR迭代。

此外，把解线性方程组的SOR迭代直接用于非线性方程组（9.1.2），在每松弛步中解一个单变量非线性方程，如用Newton法求此方程解，则可得一种以SOR为主、以Newton法为辅助迭代的SOR-Newton迭代法。

9.5.1 Newton-SOR迭代法

若记$A_k = F'(x^k)$，$b_k = F'(x^k)x^k - F(x^k)$，则（9.5.1）可改写成

9.5.2 $A_k x^{k+1} = b_k$ $(k=0,1,\cdots)$

将A_k分裂为$A_k = B_k - C_k$，其中B_k非奇，且B_k^{-1}容易计算。例如B_k为对角阵，下三角阵时，可构造迭代法：

$x^{k,j+1} = B_k^{-1}C_k x^{k,j} + B_k^{-1}b_k$ $(j=0,1,\cdots)$

记$H_k = B_k^{-1}C_k$，则上式可写成

$x^{k,j+1} = H_k x^{k,j} + B_k^{-1}b_k$ $(j=0,1,\cdots)$

若$\lim\limits_{j\to\infty} x^{k,j}$存在，在迭代$j_k$次后，如果

$\| x^{k,j_k} - x^{k+1} \| \leqslant \varepsilon$

则有$x^{k+1} \approx x^{k,j_k}$，这里$x^{k+1}$表示（9.5.2）的精确解．如果直接取

9.5.3 $x^{k+1} = x^{k,j_k} = H_k x^{k,j_k-1} + B_k^{-1} b_k$

$= H_k(H_k x^{k,j_k-2} + B_k^{-1} b_k) + B_k^{-1} bx = \cdots$

$= H_k^{j_k} x^{k,0} + (H_k^{j_k-1} + \cdots + H_k + I) B_k^{-1} b_k$

注意到

$B_k^{-1} A_k = B_k^{-1}(B_k - C_k) = I - H_k$

$(H_k^{j_k-1} + \cdots + H_k + I)(I - H_k) = I - H_k^{j_k}$

取$x^{k,0} = x^k$，由（9.5.3）可得

9.5.4 $x^{k+1} = x^k - (I - H_k^{j_k}) x^k$

$+ (H_k^{j_k-1} + \cdots + H_k + I) B_k^{-1}(A_k x^k - F(x^k))$

$= x^k - (H_k^{j_k-1} + \cdots + H_k + I) B_k^{-1} F(x^k)$ （$k = 0, 1, \cdots$）

这就是非线性—线性迭代法的一般格式．若将A_k分裂为

$A_k = D_k - L_k - U_k$

其中D_k，L_k，U_k分别为A_k的对角阵、严格下三角阵与严格上三角阵，并取$B_k = D_k$，$C_k = L_k + U_k$．当D_k非奇，则

$H_k = D_k^{-1}(L_k + U_k)$

代入（9.5.4），得Newton-Jacobi 迭代：

9.5.5 $x^{k+1} = x^k - (H_k^{j_k-1} + \cdots + I) D_k^{-1} F(x^k)$ （$k = 0, 1, \cdots$）

如果引入松弛因子$\omega_k \geqslant 0$，并取

$$B_k = \frac{1}{\omega_k}(D_k - \omega_k L_k), \quad C_k = \frac{1}{\omega_k}[(1 - \omega_k) D_k + \omega_k U_k]$$

$H_k = H_k(\omega_k) = B_k^{-1} C_k = (D_k - \omega_k L_k)^{-1}[(1 - \omega_k) D_k + \omega_k U_k]$

代入（9.5.4），则得

9.5.6 $x^{k+1} = x^k - \omega_k [H_k^{j_k-1}(\omega_k) + \cdots$

$+ H_k(\omega_k) + I](D_k - \omega_k L_k)^{-1} F(x^k)$

（$k = 0, 1, \cdots$）

这称为Newton-SOR迭代法．当$\omega_k = 1$时就是Newton-Seidel

迭代法．在N-SOR 迭代（9.5.6）中，松弛因子 ω_k 及迭代次数j_k如何取，对收敛速度及计算效率均有较大影响，从计算效率考虑要求选最佳松弛因子ω_k，使（9.5.6）生成的迭代序列$\{x^k\}$收敛最快．它与每个迭代步矩阵$F'(x^k)$的特征值有关，计算很困难，所以通常取$\omega_k=\omega$，解法与解线性方程组SOR法相同．当$0<\omega<2$时，迭代收敛，迭代次数 j_k 可选为$j_k=m$或$j_k=k+1$，$j_k=2^k$等等，也可根据$\| x^{k,j_k}-x^{k,j_k-1}\|\leq\varepsilon$来控制．若取$j_k=1$，则（9.5.6）可改写成

9.5.7 $$x^{k+1}=x^k-\omega_k(D_k-\omega_k L_k)^{-1}F(x^k) \quad (k=0,1,\cdots)$$

称为一步N-SOR迭代，它具有计算简单，计算量省的特点，但它只有线性敛速．总的说，N-SOR迭代（9.5.6）在当$\omega_k\equiv\omega$且$j_k=m$时，都只有线性敛速，只当j_k依赖k序列$\{x^k\}$时才可能具有超线性敛速．

9.5.2 非线性松弛迭代法

把解线性方程组的Gauss-Seidel迭代法的思想直接用于解非线性方程组（9.1.1），就可得到非线性松弛迭代法．设$x^k=(x_1^k,\cdots,x_n^k)^T$是方程（9.1.1）的$k$次近似解，则求$x^{k+1}$的第$i$个分量$x_i^{k+1}$，可由$F=0$的第$i$个分量方程

9.5.8 $$f_i(x_1^{k+1},\cdots,x_{i-1}^{k+1},x_i,x_{i+1}^k,\cdots,x_n^k)=0 \quad (i=1,\cdots,n)$$

的解x_i得到．由x^k计算x^{k+1}的具体过程如下：

当$i=1$时，由 $f_1(x_1,x_2^k,\cdots,x_n^k)=0$解出$x_1$，记作$x_1^{k+1}$．当$i=2$时，由 $f_2(x_1^{k+1},x_2,x_3^k,\cdots,x_n^k)=0$ 解出x_2，记作x_2^{k+1}．一般情况下，由（9.5.8）解出x_i记作x_i^{k+1}．由于每个分量方程为单变量非线性方程，所以，若用Newton法求解，就得到Gauss-Seidel-Newton 迭代．在求出（9.5.8）的解x_i后，引入松弛参数$\omega_k>0$，并令

9.5.9 $$x_i^{k+1}=x_i^k+\omega_k(x_i-x_i^k) \quad (i=1,\cdots,n)$$

则称此迭代过程为非线性SOR迭代．若求（9.5.8）的非线

性方程使用Newton法，则称此迭代为SOR-Newton法．它是双层迭代，主迭代为SOR法，内层迭代用Newton 法求n个单变量的非线性方程．若用m步Newton 法，则得m步SOR-Newton迭代．当$m=1$时，可得一步SOR-Newton迭代

9.5.10 $$x_i^{k+1}=x_i^k-\omega_k\frac{f_i(x_1^{k+1},\cdots,x_{i-1}^{k+1},x_i^k,\cdots,x_n^k)}{\partial_i f_i(x_1^{k+1},\cdots x_{i-1}^{k+1},x_i^k,\cdots x_n^k)}$$

$$(i=1,\cdots,n)$$

这个程序由x^k计算x^{k+1}只用n个分量函数值及n个分量偏导数值，相当2个函数值，且不用解线性方程组．其存储单元少，特别适用于求解大型稀疏的非线性方程组．这也是一般非线性松弛法的优点，但这类方法只有线性敛速，主要缺点是收敛慢．

9.6

9.6 割线法

为了避免在迭代过程中计算导数，可采用割线法．离散Newton 法实际上也是一种割线法．割线法是通过对函数$F(\cdot)$的线性插值导出的，是$n=1$情形的推广．对n维情形，就是对空间R^{n+1}中的n个"分量超曲面"$y=f_i(x)$（$i=1$，…，n），用x^k附近的$n+1$个点$x^{k,j}$（$j=0$，1，…，n）形成的n个"超割平面"近似代替，也就是用这$n+1$个点作为插值节点构造一个仿射映射

9.6.1 $$L_k(x)=A_kx+b_k$$

其中 $L_k(x)=(l_1(x),\cdots,l_n(x))^T,A_k\in R^{n\times n},b_k\in R^n$，使

9.6.2 $$l_i(x^{k,j})=f_i(x^{k,j})\quad(i=1,\cdots,n,j=0,1,\cdots,n)$$

然后用线性方程组$L_k(x)=0$的解x^{k+1}近似非线性方程组$F(x)=0$的解x^*．显然，必须选取这$n+1$个点$x^{k,j}$（$j=0,1$，…，n）以保证矩阵A_k非奇异．从几何观点来说，就是要求这$n+1$个点能张成空间R^n，即n个向量$x^{k,j}-x^{k,0}$（$j=1$，…，n），

必须线性无关，这时，称这$n+1$个点处于**一般位置**.

9.6.3 定理 设给定$x^{k,j}\in R^n$（$j=0,1,\cdots,n$）及$F:D\subset R^n\to R^n$则当且仅当$\{x^{k,j}\}$处于一般位置上时，存在唯一的仿射映射（9.6.1）使它满足条件（9.6.2）。此外，当且仅当$\{F(x^{k,j})\}$处于一般位置上时，矩阵A_k才非奇异.

根据这个定理，可以构造求方程（9.1.1）的一般割线法. 设$x^{k,j}$及$F(x^{k,j})$（$j=0,1,\cdots,n$），都处于一般位置上，若已知x^k为方程（9.1.1）的k次近似，取$x^{k,0}=x^k$，由条件（9.6.2）有

$$A_kx^k+b_k=F(x^k)$$

$$A_kx^{k,j}+b_k=F(x^{k,j})\quad(j=1,2,\cdots,n)$$

两式相减，得

9.6.4
$$\begin{cases}A_k(x^{k,j}-x^k)=F(x^{k,j})-F(x^k) & (j=1,\cdots,n)\\ b_k=F(x^k)-A_kx^k\end{cases}$$

记为

9.6.5
$$\begin{cases}H_k=(x^{k,1}-x^k,x^{k,2}-x^k,\cdots,x^{k,n}-x^k)\\ \Gamma_k=(F(x^{k,1})-F(x^k),F(x^{k,2})-F(x^k),\\ \qquad\cdots,F(x^{k,n})-F(x^k))\end{cases}$$

由于$\{x^{k,j}\}$在一般位置上，故H_k非奇异. 由（9.6.4）可得

9.6.6 $A_kH_k=\Gamma_k,\quad A_k=\Gamma_kH_k^{-1}$

由于$\{F(x^{k,j})\}$在一般位置上，故Γ_k非奇异，A_k非奇异，且

$$A_k^{-1}=H_k\Gamma_k^{-1}$$

于是方程（9.6.1）有解.

9.6.7 $x^{k+1}=x^k-A_k^{-1}F(x^k)$或$x^{k+1}=x^k-H_k\Gamma_k^{-1}F(x^k)$

$$(k=0,1,\cdots)$$

这就是求非线性方程组的一般割线法。割线法的计算量在很大程度上依赖于插值点的选取，选的适当可减少计算量并保证算法有较高收敛速度.

由（9.6.5）H_k的定义，可得

9.6.8 $x^{k,j}=x^k+H_k e_j \quad (j=1,\cdots,n)$

其中e_j是第j个单位坐标向量．若反过来考虑，任给一个非奇异矩阵$H_k\in \boldsymbol{R}^{n\times n}$，则由（9.6.8）可确定$n$个点，这$n$个点连同$x^k$组成一组插值点．显然，它们处于一般位置上，因此，给定的非奇异矩阵H_k等价于给定一组处于一般位置上的插值点，这样就可从构造H阵出发，形成各种割线法．

设给定x和非奇异矩阵$H\in \boldsymbol{R}^{n\times n}$，令

$$A(x,H)=\Gamma H^{-1}$$

其中

$$\Gamma=(F(x+He_1)-F(x),\cdots,F(x+He_n)-F(x))$$

则得割线法一般形式为

9.6.9 $x^{k+1}=x^k-A(x^k,H_k)^{-1}F(x^k) \quad (k=0,1,\cdots)$

如取

$$H_k=\mathrm{diag}(h_1^k,h_2^k,\cdots,h_n^k), \quad h_j^k\neq 0 \quad (j=1,\cdots,n)$$

则

$$\Gamma_k=[F(x^k+h_1^k e_1)-F(x^k),\cdots,F(x^k+h_n^k e_n)-F(x^k)]$$

9.6.10 $$A(x^k,H_k)=\Gamma_k H_k^{-1}=\left(\frac{1}{h_1^k}[F(x^k+h_1^k e_1)-F(x^k)],\cdots,\frac{1}{h_n^k}[F(x^k+h_n^k e_n)-F(x^k)]\right)$$

上式确定的$A(x^k, H_k)$正是$F(x)$的Jacobi阵$F'(x)$的离散形式，由此得到的割线法就是离散Newton法（9.3.9）．当$h_j^k=f_j(x^k)$时，即是（9.3.10）的Newton-Steffensen程序．

若取$h_j^k=x_j^{k-1}-x_j^k$ $(j=1, \cdots, n)$

即插值点只与x^{k-1}及x^k有关，则此时

9.6.11 $$A(x^k,H_k)=\left(\frac{1}{x_1^{k-1}-x_1^k}[F(x^k+(x_1^{k-1}-x_1^k)e_1)-F(x^k)],\right.$$

$$\cdots,\frac{1}{x_n^{k-1}-x_n^k}[F(x^k+(x_n^{k-1}-x_n^k)e_n)-F(x^k)]\Big)$$

对应的割线法（9.6.9）就称为两点割线法。它只要给定两个初始近似x^0及x^1，即可按（9.6.9）及（9.6.11）的格式进行迭代。若取

$$H_k=[h_1^k e_1,\ \sum_{i=1}^{2}h_i^k e_i,\cdots,\ \sum_{i=1}^{n}h_i^k e_i]=\begin{pmatrix} h_1^k & h_1^k & \cdots & h_1^k \\ & h_2^k & \cdots & h_2^k \\ & & \ddots & \vdots \\ 0 & & & h_n^k \end{pmatrix}$$

其中 $h_i^k=x_i^{k-1}-x_i^k$

则

$$\Gamma_k=\Big[F(x^k+h_1^k e_1)-Fx^k,\cdots,F(x^k+\sum_{i=1}^{n}h_i^k e_i)-F(x^k)\Big]$$

若引进矩阵

$$p=\begin{pmatrix} 1 & -1 & & & 0 \\ & 1 & -1 & & \\ & & \ddots & \ddots & \\ & & & \ddots & -1 \\ 0 & & & & 1 \end{pmatrix}$$

则

$$H_k p=\operatorname{diag}(h_1^k,\cdots,h_n^k)$$

$$\Gamma_k p=\Big[F(x^k+h_1^k e_1)-F(x^k),F(x^k+\sum_{i=1}^{2}h_i^k e_k)$$

$$-F(x^k+h_1^k e_1),\cdots,F(x^k+\sum_{i=1}^{n}h_i^k e_i)-F(x^k+\sum_{i=1}^{n-1}h_i^k e_i)\Big]$$

于是

9.6.12 $A(x^k,H^k)=\Gamma_k H_k^{-1}=(\Gamma_k p)(H_k p)^{-1}$

$$=\Big[\frac{1}{h_1^k}(F(x^k+h_1^k e_1)-F(x^k)),\cdots,\frac{1}{h_n}(F(x^k+\sum_{i=1}^{n}h_i^k e_i)-F(x^k+\sum_{i=1}^{n-1}h_i^k e_i))\Big]$$

这样得到的割线法（9.6.9）与（9.6.12）是另一种形式的两点割线法，它每步也要算n个函数值（即n^2个分量函数值）。对一点割线法即离散Newton法，适当选取 h_j^k $(j=1,\cdots,n)$，生成的序列$\{x^k\}$可达到平方敛速。对两点割线法则有以下收敛定理。

9.6.13 **定理** 假定$F:D\subset R^n\to R^n$在x^* 的开邻域$S_0=S(x^*,\delta_0)\subset D$连续可导，$F(x^*)=0$，且$F'(x^*)$非奇异，则存在球$S=S(x^*,\delta)\subset S_0$，使对任何$x^0$，$x^1\in S$，由两点割线法生成的迭代序列$\{x^k\}\subset S$，且超线性收敛于$x^*$。如果存在$\gamma>0$，使

9.6.14 $\|F'(x)-F'(y)\|\leqslant\gamma\|x-y\|,\quad\forall x,y\in S_0$

成立，则此序列$\{x^k\}$的收敛阶$p_R\geqslant\tau_1=\dfrac{1+\sqrt{5}}{2}\approx1.618$，这里$\tau_1$是方程$t^2-t-1=0$的唯一正根。

由此可知，两点割线法的效率

$$e(S_1)=\frac{\ln\left(\frac{1+\sqrt{5}}{2}\right)}{n}$$

当$n\geqslant3$时，

$$e(S_1)<e(N_1)=\frac{\ln2}{n+1}$$

这里$e(N_1)$是Newton法效率。为提高两点割线法效率，可对它使用Шаманский技巧，这就是Brent 方法的另一种算法。只要由x^{k-1}及x^k确定一个非奇阵H_k，就可得到两点割线法，

现要求H_k为正交阵，取$H_k=h_kQ_k$，其中Q_k是由向量$x^{k-1}-x^k$找到的一个镜像变换阵，使

$$x^{k-1}-x^k=h_kQ_ke_1$$

其中$h_k=\| x^{k-1}-x^k \|_2$。这是容易做到的，事实上，对任意向量$x\in \boldsymbol{R}^n$。记$\sigma=\pm\| x \|_2$，设$x\neq-\sigma e_1$，则必存在一个镜像变换阵U，使$Ux=-\sigma e_1$。

可以证明

$$U=I-2\frac{uu^T}{\| u \|_2^2}$$

其中$u=x+\sigma e_1$。

根据这一性质，由$x^{k-1}-x^k$可求得

$$Q_k=I-2\frac{u_ku_k^T}{\| u_k \|_2^2}$$

其中 $u_k=x^{k-1}-x^k\pm h_ke_1$，当$x_1^{k-1}-x_1^k\geqslant 0$时取'+'号。否则取'-'号，于是可得下列算法：

9.6.15 算法 给定$F:D\subset\boldsymbol{R}^n\to\boldsymbol{R}^n$及两个不同初始近似$x^0$，$\overline{x^0}\in D$，以及$k_0\geqslant 1$，然后按下列步骤求解：

1° 确定正交矩阵Q_k，使

$$\overline{x}^k=x^k+h_kQ_ke_1 \qquad (h_k=\| x^k-x^{k-1} \|_2)$$

2° 计算矩阵

$$\Gamma_k=\frac{1}{h_k}[F(x^k+h_kQ_ke_1)-F(x^k),\cdots,F(x^k+h_kQ_ke_n)-F(x^k)]$$

（注意：这里$F(\overline{x}^k)=F(x^k+h_kQ_ke_1)$已知，故可节省计算$F$值一次）。

3° 以$A_k=\Gamma_kQ_k$进行k_0次迭代计算，即令$y^{k,0}=x^k$并计算

$$y^{k,l}=y^{k,l-1}-A_k^{-1}F(y^{k,l-1}),\quad (l=1,\cdots,k_0)$$

4° 置$x^{k+1}=y^{k,k_0}$，$\overline{x}^{k+1}=y^{k,k_0-1}$，若$x^{k+1}$满足精度要求，则置$x^*=x^{k+1}$，并停止迭代，否则执行5°。

5° 置$k=k+1$，转1°。

以上算法称为Brent的S_{k_0}算法。当$k_0=1$即为两点割线法(9.6.9)与(9.6.11)。S_{k_0}算法每迭代一步需计算$n+k_0-1$次函数值。

在定理(9.6.13)的条件下，S_{k_0}算法生成的序列$\{x^k\}$收敛于x^*，其收敛阶至少为

$$p(k_0)=\frac{k_0+\sqrt{k_0^2+4}}{2}$$

由以上结果可知，S_{k_0}算法的效率为

$$e(S_{k_0})=\frac{\ln p(k_0)}{n+k_0-1}$$

对固定的n，可求最优的$\overline{k_0}$，使

$$e(S_{\overline{k}_0})=\max_{k_0}\frac{\ln p(k_0)}{n+k_0-1}$$

下表列出某些固定n时，$\overline{k_0}$及效率比$\frac{e(S_{\overline{k}_0})}{e(N_1)}$的值。

9.6.16 表

n	2	3	5	10	20	50	100	1000
$\overline{k_0}$	3	4	5	8	12	23	38	226
$e(S_{\overline{k}_0})/e(N_1)$	1.29	1.39	1.58	1.96	2.44	3.21	3.87	6.39

9.7

9.7 拟Newton法

9.7.1 拟Newton法的基本思想

Newton 法，离散Newton 法及割线法都可表示为

9.7.1　$x^{k+1}=x^k-A_k^{-1}F(x^k)\quad(k=0,1,\cdots)$

它们的计算量较大，这使得如何减少计算量并保持较高的收敛速度成为求解非线性方程组一个十分重要的问题。拟Newton 法就是针对这一问题提出的新方法，它用A_k近似$F'(x^k)$，而A_{k+1}可在A_k的基础上用一个低秩的矩阵ΔA_k来校正，这样既减少了计算函数值次数，又使求A_k^{-1}运算量降为$O(n^2)$次算术运算。

设$F:D\subset R^n\to R^n$连续可导，$s_k=x^{k+1}-x^k\neq 0$，x^k，$x^{k+1}\in D$，给定$\varepsilon>0$，$\exists\delta>0$，当$\|s_k\|<\delta$时

$$\|F(x^k)-F(x^{k+1})-F'(x^{k+1})(x^k-x^{k+1})\|\leqslant\varepsilon\|s_k\|$$

成立。因此有

$$F(x^k)\approx Fx(^{k+1})+F'(x^{k+1})(x^k-x^{k+1})$$

其近似程度随$\|x^k-x^{k+1}\|$减少而增大。若以A_{k+1}近似代替$F'(x^{k+1})$，则要求A_{k+1}满足方程

9.7.2　$A_{k+1}(x^{k+1}-x^k)=F(x^{k+1})-F(x^k)$

当$n=1$时，（9.7.2）为

$$A_{k+1}=\frac{F(x^{k+1})-F(x^k)}{x^{k+1}-x^k}$$

它是$F(x)$关于点x^{k+1}与x^k的差商。当$n>1$时表明矩阵A_{k+1}关于点x^k与x^{k+1}具有“差商”性质，称（9.7.2)为广义差商条件。因为A_{k+1}可以看作$F'(x^{k+1})$的近似矩阵，故（9.7.2)又称为拟Newton方程。由于A_{k+1}有n^2个元素，而方程（9.7.2）只给出n个条件，所以当$n>1$时，A_{k+1}不能完全确定，它还有很大的选择余地。如果A_{k+1}能表示为$A_k+\Delta A_k$（ΔA_k为低秩矩阵），则可得到一类迭代算法：

9.7.3
$$\begin{cases}x^{k+1}=x^k-A_k^{-1}F(x^k)\\ A_{k+1}(x^{k+1}-x^k)=F(x^{k+1})-F(x^k)\quad(k=0,1,\cdots)\\ A_{k+1}=A_k+\Delta A_k\ ,\ \mathrm{rank}\Delta A_k=m\geqslant 1\end{cases}$$

称为拟Newton法，或称为秩m校正方法。如果对所有k，$A_k^{-1}=H_k$存在，则迭代（9.7.3）是完全确定的。利用Sherman-Morrison-Woodbury公式（9.7.4），可得到ΔH_k的一个显式表达式。

Sherman-Morrison-Woodbury（谢门-莫里森-伍德伯依）公式如下：设$A\in R^{n\times n}$非奇异，U，$V\in R^{n\times m}$，$m\leqslant n$，则当且仅当$I+V^TA^{-1}U$非奇异时，$A+UV^T$也非奇异，且

9.7.4 $$(A+UV^T)^{-1}=A^{-1}-A^{-1}U(I+V^TA^{-1}U)^{-1}V^TA^{-1}$$

因为$\operatorname{rank}(\Delta A_k)=m$，而任何秩$m$的$n$阶矩阵都可分解为

$$\Delta A_k=U_kV_k^T,\quad U_k,V_k\in R^{n\times m},\quad \operatorname{rank}U_k=\operatorname{rank}V_k=m$$

所以，如果$(I+V_k^TA_k^{-1}U_k)$非奇，则

$$\begin{aligned}H_{k+1}&=(A_k+\Delta A_k)^{-1}\\&=A_k^{-1}-A_k^{-1}U_k(I+V_k^TA_k^{-1}U_k)^{-1}V_k^TA_k^{-1}\\&=H_k+\Delta H_k\end{aligned}$$

其中

$$\Delta H_k=Y_kW_k^T$$

$$Y_k=-A_k^{-1}U_k(I+V_k^TA_k^{-1}U_k)^{-1},\quad W_k=(A_k^{-1})^TV_k$$

显然，Y_k，$W_k\in R^{n\times m}$，且它们的秩为m，故ΔH_k仍为一个秩m矩阵。这样，（9.7.3）又可写成

9.7.5 $$\begin{cases}x^{k+1}=x^k-H_kF(x^k)\\H_{k+1}[F(x^{k+1})-F(x^k)]=x^{k+1}-x^k\quad(k=0,1,\cdots)\\H_{k+1}=H_k+\Delta H_k,\quad \operatorname{rank}\Delta H_k=m\end{cases}$$

这就是逆秩m校正公式。

不同的计算方案对应不同算法，常用的算法只是$m=1$及$m=2$的公式，即使用秩1及秩2校正公式。

9.7.2 Broyden方法

Broyden 于1965年首先提出了一种秩1的校正公式：由

x^k计算x^{k+1}只计算一次F函数值及做$O(n^2)$次算术运算．为书写方便，把上下标去掉，用$\overline{x}$表示由x出发得到的新近似，相应地，由A得到$\overline{A}$，H得到$\overline{H}$，迭代过程（9.7.1）可写成

$\overline{x}=x-A^{-1}F(x)$

拟Newton方程（9.7.2）可写成

9.7.6 $\overline{A}s=y$

其中$s=\overline{x}-x$，$y=F(\overline{x})-F(x)$。Broyden 方法要求$\overline{A}$与A在向量s的正交补$s^{\perp}$上两者无任何差别，即对$\forall z\in s^{\perp}$，$\overline{A}z=Az$，也就是

$(\overline{A}-A)z=0$

对$\forall(z, s)=s^Tz=0$成立，故$\overline{A}-A$为秩1矩阵．可设

$\overline{A}-A=us^T$

这里$u\in \boldsymbol{R}^n$为待定向量．代入上式则得

$us^Tz=0$

由（9.7.6）可得

$(\overline{A}-A)s=y-As$

于是

$us^Ts=y-As$

$$u=\frac{y-As}{s^Ts}$$

从而推得

9.7.7 $\overline{A}=A+\dfrac{(y-As)s^T}{s^Ts}, \qquad s\neq 0$

当$s=0$时，$\overline{x}=x$，迭代即终止．(9.7.7)就是秩1校正公式，这个公式具有一个重要性质，即在所有满足拟Newton方程的矩阵中，$\overline{A}$是在Frobenius（佛罗比尼乌斯）范数意义下最接近A的矩阵．

9.7.8 **定理** 设给定$A\in \boldsymbol{R}^{n\times n}$，$y\in \boldsymbol{R}^n$和非零向量$s\in \boldsymbol{R}^n$，则由(9.7.7)确定的矩阵$\overline{A}$是以下问题的唯一解

$$\min\{\|\hat{A}-A\|_F | \hat{A}s=y\}$$

由公式（9.7.7）与拟Newton 法（9.7.1）得

9.7.9
$$\begin{cases} x^{k+1}=x^k-A_k^{-1}F(x^k) \\ A_{k+1}=A_k+\dfrac{(y^k-A_ks_k)s_k{}^T}{s_k{}^Ts_k} \quad (k=0,1,\cdots) \end{cases}$$

其中$s_k=x^{k+1}-x^k$，$y^k=F(x^{k+1})-F(x^k)$。公式（9.7.9）称为Broyden 方法。

满足拟Newton 方程（9.7.6）的秩1算法有很多不同选法，如果要求$\overline{A}$与A在向量c的正交补$c^\perp$上两者无任何差别，则可得$\Delta A=\overline{A}-A=uc^T$，由

$$(\overline{A}-A)s=y-As=F(\overline{x})$$

可得 $uc^Ts=F(\overline{x})$，即$u=\dfrac{F(\overline{x})}{c^Ts}$，于是

9.7.10
$$\overline{A}=A+\frac{F(\overline{x})c^T}{c^Ts}$$

在这公式中取不同的c，即可得到不同的秩1校正公式。若$c=s$，则可得（9.7.1）；若取$c=F(\overline{x})=F(x^{k+1})$，则可得另一秩1拟 Newton法，

9.7.11
$$\begin{cases} x^{k+1}=x^k-A_k^{-1}F(x^k) \\ A_{k+1}=A_k+\dfrac{F(x^{k+1})F(x^{k+1})^T}{F(x^{k+1})^T(x^{k+1}-x^k)} \end{cases}$$

这称为Broyden 第二方法。其特点是ΔA_k为对称矩阵，因此，它适合于对F的Jacobi矩阵为对称的方程组求解，这时初始矩阵A_0也应取为对称矩阵。

由此得到的 Broyden 方法（9.7.9）及（9.7.11），计算都较简单，对给定的x^k及A_k，每步只需算一个新的F函数值，然后求解$A_ks_k=-F(x^k)$得到x^{k+1}，再计算Fx^{k+1}，从而求得A_{k+1}。表面上看，解方程要用$O(n^3)$的算术运算，但如

通过QR分解，可使运算量降为$O(n^2)$. 通常可设$A_k=Q_kR_k$，其中Q_k为正交阵，R_k为上三角阵，由此形成A_{k+1}的QR分解，只需$O(n^2)$算术运算.

对校正公式（9.7.7），利用 Sherman-Morrison公式，有

$$(A+uv^T)^{-1}=A^{-1}-\frac{1}{\sigma}(A^{-1}uv^TA^{-1})$$

其中$A\in R^{n\times n}$非奇，$u、v\in R^n$，$\sigma=1+v^TA^{-1}u\neq 0$.

令$H=A^{-1}, \overline{H}=\overline{A}^{-1}$，$v=s$，$u=\frac{y-As}{s^Ts}$，则得

9.7.12 $$\overline{H}=H-\frac{Hus^TH}{1+s^THu}=H+\frac{(s-Hy)s^TH}{s^THy}$$

其中$s^THy\neq 0$，这是$\overline{H}$非奇的充要条件. 由（9.7.12）则可得到逆 Broyden 秩1公式

9.7.13 $$\begin{cases} x^{k+1}=x^k-H_kF(x^k) \\ H_{k+1}=H_k+\dfrac{(s_k-H_ky^k)s_k{}^TH_k}{s_k{}^TH_ky^k} \end{cases} \quad (k=0,1,\cdots)$$

对应于（9.7.11）的逆Broyden秩1第二公式为

9.7.14 $$\begin{cases} x^{k+1}=x^k-H_kF(x^k) \\ H_{k+1}=H_k+(s_k-H_ky^k)\dfrac{(s_k-H_ky^k)^T}{(s_k-H_ky^k)^Ty^k} \end{cases} \quad (k=0,1,\cdots)$$

在一定条件下，可以证明Broyden 秩1方法生成的序列$\{x^k\}$是局部超线性收敛的.

9.7.15 **例** 用 Broyden 方法（9.7.13）求解方程组

$$\begin{cases} f_1(x_1,x_2)=x_1{}^2-x_2-1=0 \\ f_2(x_1,x_2)=(x_1-2)^2+(x_2-0.5)^2-1=0 \end{cases}$$

初始近似取为$x^0=(0, 0)^T$。

解：先计算$F(x^0)=(-1, 3.25)^T$. 因为

$$F'(x)=\begin{bmatrix} 2x_1 & -1 \\ 2x_1-4 & 2x_2-1 \end{bmatrix}$$

故 $$F'(x^0)=\begin{bmatrix} 0 & -1 \\ -4 & -1 \end{bmatrix}$$

$$H^0=F'(x^0)^{-1}=\begin{bmatrix} 0.25 & -0.25 \\ -1 & 0 \end{bmatrix}$$

由公式 (9.7.13) 可算得

$s_0=(1.0625, -1)^T$, $x^1=(1.0625, -1)^T$

$F(x^1)=(1.12890625, 2.12890625)^T$

$$y^1=F(x^1)-F(x^0)=\begin{bmatrix} 2.12890625 \\ -1.12109375 \end{bmatrix}$$

$$H_1=\begin{bmatrix} 0.3557441 & -0.2721932 \\ -0.5224991 & -0.1002162 \end{bmatrix}$$

由此再算得：

$x^2=(1.240372062, -0.196797467)^T$

按上述算法，经十一次迭代可得到具有 12 位有效数字的近似解

$x_{11}=(1.54634288332, 1.39117631279)^T$

如用 Newton 法，则只要七次迭代即可达到同样精度的近似解。 Broyden 方法虽然比 Newton 迭代多四次，但它每步计算量比 Newton 法少得多，所以，单从理论上还不能断定这两种方法效率的高低。

9.7.3 秩2校正公式

在 (9.7.3) 与 (9.7.5) 式中，若取$m=2$，则可得到一

系列公式，这就是秩2校正公式。在计算中，通常以逆拟Newton 公式（9.7.5）使用较多，为了具体得到 ΔH_k 表达式，设 y_k，$w_k \in \mathbf{R}^{n\times 2}$，并分别表示为

$y_k=[y_k^1,\ y_k^2]$ 及 $w_k=[w_k^1,\ w_k^2]$

其中 y_k^i，$w_k^i \in \mathbf{R}^n$，$(i=1,\ 2)$。将它代入逆拟 Newton 公式

9.7.16 $H_{k+1}y^k=s_k$

由于 $\Delta H_k=y_k w_k^T$，则（9.7.16）可写成

9.7.17 $(H_k+y_k^1(w_k^1)^T+y_k^2(w_k^2)^T)y^k=s_k$

若记 $r_i=(w_k^i)^T y^k \quad (i=1,\ 2)$，则（9.7.17）可写成

$r_1 y_k^1+r_2 y_k^2=s_k-H_k y^k$

当 $r_i\neq 0 \quad (i=\ 1,\ 2)$ 时，可取

$$y_k^1=\left(\frac{1}{r_1}\right)s_k,\qquad y_k^2=-\left(\frac{1}{r_2}\right)H_k y^k$$

显然这时（9.7.16）满足，于是得

9.7.18 $$H_{k+1}=H_k+\frac{s_k(w_k^1)^T}{(w_k^1)^T y^k}-\frac{H_k y^k(w_k^2)^T}{(w_k^2)^T y^k}\quad (k=0,1,\cdots)$$

通过计算表明，当且仅当

$(w_k^2)^T F(x^k)\neq 0 \quad (k=0,1,\cdots)$

时，非奇条件 $(I+w_k^T A_k y_k)^{-1}$ 成立。因此，只要（w_k^i）满足

$(w_k^i)^T y^k\neq 0 \qquad (i=1,2)$

$(w_k^2)^T F(x^k)\neq 0 \qquad (k=0,1,\cdots)$

算法（9.7.18）则生成一个非奇矩阵序列 $\{H_k\}$，它满足方程 $(H_{k+1}-H_k)y^k=-H_k F(x^{k+1})$，因此，只要选择不同的 w_k^1 及 w_k^2 就可产生不同的校正公式。当 w_k^1 及 w_k^2 线性相关，则得秩1校正公式；当 w_k^1 及 w_k^2 线性无关，则可得秩 2 校正公式。通常，较好的秩2算法有如下几种：

1° D-F-P(Davidon-Fletcher-Powell)(达维东-弗莱特彻-保韦尔)秩2算法。在（9.7.18）中取

$w_k^1=s_k$, $w_k^2=H_k^T y^k \quad (k=0,1,\cdots)$

可得D-F-P算法计算公式

9.7.19
$$\begin{cases} x^{k+1}=x^k-H_kF(x^k) \\ H_{k+1}=H_k+\dfrac{s_k s_k^T}{(s_k)^T y^k}+\dfrac{H_k y^k (y^k)^T H_k}{(y^k)^T H_k y^k} \quad (k=0,1,\cdots) \end{cases}$$

2° B-F-S(Broyden-Fletcher-Shanno)(布罗依登-弗莱特彻-香诺)秩2算法．在（9.7.18）中，取

$$\frac{(w_k^1)^T}{(w_k^1)^T y^k}=\mu_k\frac{s_k^T}{s_k^T y^k}-\frac{(y^k)^T H_k}{s_k^T y^k},\ w_k^2=s_k$$

其中 $\mu_k=1+\dfrac{(y^k)^T H_k y^k}{s_k^T y^k}\in \boldsymbol{R}^1$，则得$B$-$F$-$S$算法公式

9.7.20
$$\begin{cases} x^{k+1}=x^k-H_kF(x^k) \\ H_{k+1}=H_k+\dfrac{\mu_k s_k s_k^T-s_k(y^k)^T H_k-H_k y^k s_k^T}{s_k^T y^k} \end{cases}$$

这公式比（9.7.16）有更好的稳定性，实际计算表明它是拟Newton法中较为成功的算法．

9.8

9.8 极小化方法

求解非线性方程组（9.1.1）的问题，可以化为求多元函数的极小化问题，例如令

9.8.1
$$\phi(x)=F(x)^T F(x)=\sum_{i=1}^{n} f_i^2(x)$$

ϕ: $\boldsymbol{R}^n\to\boldsymbol{R}^1$，则 $F(x)=0$ 便是 $\phi(x)=0$ 的充分必要条件．此外，对一切 $x\in\boldsymbol{R}^n$，恒有

$\phi(x)\geqslant 0$

所以，$\min\phi(x)\geqslant 0$，即：使 $\phi(x)=0$ 的任何极小点 x^* 同时也是方程组（9.1.1）的解．这样，方程组 $F(x)=0$ 的求解问题，便化为求函数 $\phi(x)$的极小点问题．

9.8.1 下降算法

一类非常广泛和重要的极小化算法是每一次迭代都使函数值 $\phi(x^k)$减小，即

9.8.2 $\phi(x^{k+1})<\phi(x^k)\quad(k=0,1,\cdots)$

这种算法称为下降算法。其一般原则是从某一点x^0出发，沿一个使 $\phi(x)$下降的方向 p_0，令

$x^1=x^0+\mu p_0,\ \mu>0$

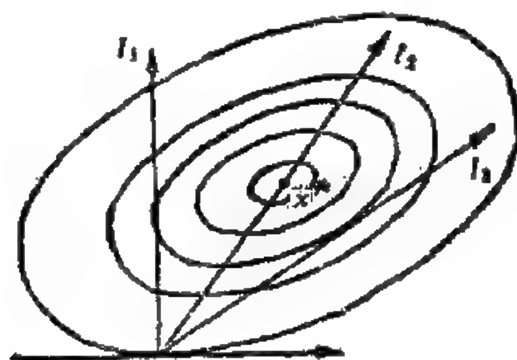
然后确定步长因子 $\mu=\mu_0$，使 $\phi(x^1)<\phi(x^0)$。一般情况是从 x^k 出发，沿 $\phi(x)$下降方向 p_k求出 x^{k+1}，即

9.8.3 $x^{k+1}=x^k+\mu_k p^k\quad(k=0,1,\cdots)$

使（9.8.2）满足，直到 $\phi(x^m)<\varepsilon$（给定精度），x^m 即为（9.1.1）的近似解。

从上述算法的一般原则可以看到：下降方向 p_k和步长因子 μ_k 构成每一迭代的修正量，这表明它们是决定算法好坏的重要因素。关于下降方向 p_k 的选择，以指向极小问题的最优解或使 ϕ 下降最快的方向为佳。以二维情形为例，若 ϕ 的等高线族如图（9.8.4）所示，x^*为极小点，图中给出三个方向 l_1，l_2，l_3，显然，最理想的是取$p_k=l_2$，因它直指极小点方向。但是，一般是难于求得的，通常可根据函数 ϕ的解析性质确定尽可能好的下降方向p_k。

9.8.4 图



步长因子μ_k的一种选择是使它满足

$$\phi(x^k+\mu_k p_k)<\phi(x^k)$$

另一种选择是使$\phi(x)$从x^k出发沿p_k方向取极小值，即

9.8.5 $$\phi(x^k+\mu_k p_k)=\min_{\mu}\phi(x^k+\mu p_k)$$

9.8.2 最速下降法

具体的极小化算法很多，最古老和基本的数值方法是最速下降法。假定$\phi(x)$二次连续可微，设x^0为初始近似，x^k为k次近似，将$\phi(x)$在x^k处按Taylor展开

9.8.6 $$\phi(x^k+\mu p_k)=\phi(x^k)+\mu\nabla\phi(x^k)^T p_k+o(\|\mu p_k\|)$$

其中$\nabla\phi(x^k)$为函数$\phi(x)$在x^k处的梯度向量。$o(\|\mu p_k\|)$对充分小的μ是高阶小量，因此，要使不等式

$$\phi(x^k+\mu p_k)<\phi(x^k)$$

成立，(9.8.6) 中μ的一次项的系数起决定作用，即忽略μ的高阶项，要求

$$\nabla\phi(x^k)^T p_k<0$$

如果$\nabla\phi(x^k)\neq 0$，则使上式成立的p_k可以有无穷多个，但使$|\nabla\phi(x^k)^T p_k|$取最大值的p_k却只有一个，因为使

$$|\nabla\phi(x^k)^T p_k|\leqslant\|\nabla\phi(x^k)\|\,\|p_k\|$$

成为等式，当且仅当$p_k=\nabla\phi(x^k)$时才可能。此时

$$\nabla\phi(x^k)^T\nabla\phi(x^k)=\|\nabla\phi(x^k)\|^2$$

亦即当$p_k=-\nabla\phi(x^k)$时使$\nabla\phi(x^k)p_k$取到负最小值，这说明在x^k的适当小范围内，负梯度方向

$$p_k=-\nabla\phi(x^k)$$

是使$\phi(x)$下降最快的方向。此时，有

9.8.7 $$x^{k+1}=x^k-\mu_k\nabla\phi(x^k)\qquad(k=0,1,\cdots)$$

其中μ_k满足 (9.8.5) 的要求，这样得到的算法就是**最速下降法**。这个方法优点是程序简单，每步迭代计算量少，从一个不太好的初始近似x^0出发也能收敛到局部极小点，但是

这个方法收敛慢，特别是 ϕ的 Hesse矩阵 $H(x)$呈病态 情形时，收敛更慢，甚至花大量时间仍算不出要求的结果。

9.8.8 定理 设 $\phi: D\subset \boldsymbol{R}^n \to \boldsymbol{R}^1$ 于凸集$D_0\subset D$上二次连续可微，其 Hesse 矩阵 $H(x)$在 D_0 上满足

$$m\ \|y\|^2 \leqslant y^T H(x) y \leqslant M\ \|y\|^2$$

其中 $M\geqslant m>0$，$y\in \boldsymbol{R}^n$ 为任意非零向量，对任给的初始近似 $x^0\in D_0$，集合 $S=\{x|\ \phi(x)\leqslant\phi(x^0)\}\subset D_0$ 有界，序列$\{x^k\}$由最速下降法产生，则

1° ϕ 在 D_0 中存在唯一的全局极小点 $x^*\in D_0$；

2° $\{x^k\}$收敛到x^*，且有估计式

$$\|x^k - x^*\| \leqslant c_2 q_0^{k/2},\quad c_2<\infty$$

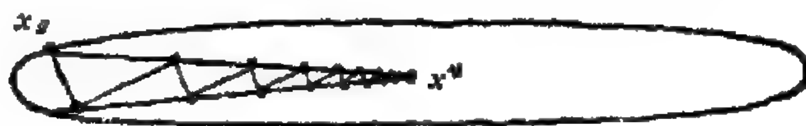
$$\phi(x^k)-\phi(x^*)\leqslant q_0^k(\phi(x^0)-\phi(x^*))$$

其中

$$q_0=(1-\frac{m^2}{M^2})<1$$

最速下降法只有线性敛速，其收敛因子 q_0 依赖于矩阵$H(x)$的条件数$\rho=\dfrac{M}{m}$（M，m 分别表示$H(x)$的最大与最小特征值）.$q_0=1-\rho^{-2}$，要使 q_0 尽量小则要求 $\rho\approx1$；但是，若遇到 ρ 很大如 $\rho^{-1}\approx0$ 的情形，则 $q_0\approx1$，此时 $H(x)$的最大与最小特征值相差很大，矩阵 $H(x)$出现严重病态. 对 ϕ 是二维正定二次函数而言，若其等高线族是一族很扁的椭圆，由最速下降法所得迭代序列$\{x^k\}$绕道前进，每次搜索方向严重偏离椭圆族中心x^*的方向，且每次迭代所跨的步长很小，如图（9.8.9），这时使用最速下降法效果极差.

9.8.9 图



9.8.3 共轭梯度法

由于最速下降方向的局部特性，一般情况下收敛速度并不理想。当 x^0 远离 x^* 时，目标函数 ϕ 下降较快，但当 x^k 进入极小点附近区域时，绕道现象严重，收敛很慢，这时应使用其它收敛快的方法。例如，用共轭梯度法或 Newton法。

共轭梯度法是基于对函数 ϕ 的二次逼近，当 ϕ 满足定理（9.8.8）的条件时，取初始近似 $x^0\in D_0$，$p_0=-\nabla\phi(x^0)$，将 $\phi(x^0+\mu p_0)$ 于 x^0 附近按 Taylor 展开，忽略三次项，得

$$\phi(x^0+\mu p_0)$$

$$\approx\phi(x^0)+\mu\nabla\phi(x^0)^T p_0+\frac{1}{2}\mu^2 p_0{}^T H(x^0)p_0$$

近似式右端为 μ 的二次函数，因 $p_0{}^T H(x^0)p_0>0$，故使该二次函数取极小的 μ 为

$$\mu_0=-\frac{\nabla\phi(x^0)p_0}{p_0{}^T H(x^0)p_0}$$

若令 $x^1=x^0+\mu_0 p_0$，$p_1=-\nabla\phi(x^1)+\nu_0 p_0$ 要求满足 $p_1{}^T H(x^0)p_0=0$，则得

$$\nu_0=\frac{\nabla\phi(x^1)^T H(x^0)p_0}{p_0{}^T H(x^0)p_0}$$

若已求得 x^0，x^1，…，x^k 及 p_0，p_1，…，p_k，则一般地共轭梯度法计算公式为

$$\textbf{9.8.10}\quad\begin{cases}x^{k+1}=x^k+\mu_k p_k\\ \mu_k=-\dfrac{\nabla\phi(x^k)^T p_k}{p_k^T H(x^k)p_k}\ \text{或}\ \phi(x^k+\mu_k p_k)=\min\limits_{\mu}\ \phi(x^k+\mu p_k)\\ p^{k+1}=-\nabla\phi(x^{k+1})+\nu_k p_k\\ \nu_k=\dfrac{\nabla\phi(x^{k+1})^T H(x^k)p^k}{p_k^T H(x^k)p_k}\ \text{或}\ \nu_k=\dfrac{\nabla\phi(x^{k+1})^T\nabla\phi(x^{k+1})}{\nabla\phi(x^k)^T\nabla\phi(x^k)}\end{cases}$$

$$(k=0,1,\cdots)$$

在下降法中 p_k 取为 Newton 方向

9.8.11 $p_k=-F'(x^k)^{-1}F(x^k)$

时，由于 $\nabla\phi(x^k)=2F'(x^k)^TF(x^k)$，于是

$$\nabla\phi(x^k)^Tp_k=-2F(x^k)^TF'(x^k)F'(x^k)^{-1}F(x^k)$$
$$=-\|F(x^k)\|_2^2<0$$

这表明Newton方向（9.8.11）也是一种下降方向，因此Newton下山法（9.3.17）也是下降算法。在x^k靠近x^*时，取$\mu_k=1$，就是Newton法，它可用来改进最速下降的收敛性。另外，SOR-Newton法（9.5.10）是一种沿坐标方向的下降法。

9.9

9.9 延拓法

对于大多数迭代法，一般都要求初始近似x^0与方程$F(x)=0$的解x^*足够近时才收敛于x^*，但在实际计算时要找到这样的x^0是很困难的，**延拓法**(Continuation)可看作是一种扩大收敛域，求与x^*足够近的初始近似x^0的方法。

用延拓法求方程组（9.1.2）的解是引入参数$t\in R^1$，一般取$t\in[0, 1]$，并构造同伦$H:D\times[0, 1]\subset \boldsymbol{R}^n\times\boldsymbol{R}^1\to R^n$来代替映射$F$，这簇映射$H$满足

9.9.1 $H(x,0)=F_0(x), \qquad H(x,1)=F(x), \ \forall x\in D$

其中$H(x, 0)=0$的解x^0是已知的，方程$H(x, 1)=0$是要求解的问题。如果方程

9.9.2 $H(x,t)=0, \qquad t\in[0,1]$

有解$x=x(t)$，$x:[0, 1]\to\boldsymbol{R}^n$连续依赖于$t$，当$t=1$时，$x(1)=x^*$即为方程（9.1.2）的解，亦即若有连续映射$x:[0, 1]\to D\subset\boldsymbol{R}^n$使得

9.9.3 $H(x(t),t)=0, \qquad \forall t\in[0,1]$

则$x=x(t)$表示$\boldsymbol{R}^n$内一条空间曲线，它的一端表示某个给定的点x^0，另一端是$F(x)=0$的解$x^*=x(1)$，映射$H:D\times$

$[0,1]\subset \boldsymbol{R}^{n+1}\to\boldsymbol{R}^n$ 称为**同伦**(Homotopy)**映射**.延拓法就是把求解方程（9.1.2）的问题转化为求同伦方程（9.9.2）的解，故又称为**同伦算法**。因为延拓法是对原方程嵌入参数 t 而得到的，故又称为**嵌入法**(Embedding Method)。

同伦映射H可用各种不同方法构造，但它必须满足条件(9.9.1).例如，可取

9.9.4 $H(x,t)=tF(x)+(1-t)F_0(x),\qquad x\in D,t\in[0,1]$

其中 F_0 是已知映射，且 $F_0(x)=0$ 的解 x^0 是给定的。若是取 $F_0(x)=A(x-x^0)$，$A\in\boldsymbol{R}^{n\times n}$ 非奇异，则得

9.9.5 $H(x,t)=tF(x)+(1-t)A(x-x^0),x\in D,t\in[0,1]$

H还可取为

9.9.6 $H(x,t)=F(x)-(1-t)F(x^0),\ x\in D,t\in[0,1]$

若令 $t=1-e^{-s}$，由（9.9.6）可得

$H(x,s)=F(x)-e^{-s}F(x^0),\qquad x\in D,S\in[0,\infty)$

当$s=0,H(x,0)=F(x)-F(x^0)=0$时有解x^0；当$s\to\infty$，$H(x,s)\to F(x)$时满足条件（9.9.1)，故$\lim\limits_{s\to\infty}x(s)=x^*$ 是方程 $F(x)=0$ 的解。

9.9.1 数值延拓法

假定 $H:D\times[0,1]\subset\boldsymbol{R}^{n+1}\to\boldsymbol{R}^n$是满足（9.9.1）的已知同伦，若存在一个连续映射 $x:[0,1]\to D$,使(9.9.3)成立，为了求 $x^*=x(1)$，可将区间$[0,1]$分划为

9.9.7 $0=t_0<t_1<\cdots<t_N=1$

用某种迭代法求方程

9.9.8 $H(x,t_i)=0\qquad(i=1,\cdots,N)$

的解 x^i 时，用第$(i-1)$个方程的解 x^{i-1} 作初始近似，如果 t_i-t_{i-1} 充分小,则可以期望 x^{i-1} 是 x^i 的一个足够好的近似，从而使迭代法收敛。由此逐步求得 $H(x,1)=0$的解 x^N，即为方程（9.1.2）的近似解.对（9.9.8）第 i 个方程求解时，

只要用有限步迭代即可，故可取 $m_i \geqslant 1$ 步迭代.如用Newton迭代求解方程（9.9.8），则得数值延拓法序列

9.9.9
$$\begin{cases} x^{i,j+1}=x^{i,j}-[H_x(x^{i,j},t_i)]^{-1}H(x^{i,j},t_i) \\ x^{1,0}=x^0, x^{i+1,0}=x^{i,m_i} \\ \qquad\qquad (j=0,1,\cdots,\ m_i-1;\quad i=1,\cdots,N-1) \\ x^{k+1}=x^k-H_x(x^k,1)^{-1}H(x^k,1) \qquad (k=N,N+1,\cdots) \\ x^N=x^{N,0} \end{cases}$$

在一定条件下，存在[0,1]的一个分划（9.9.7）和整数序列$\{m_j,\ j=1,\cdots,N-1\}$，使$\{x^{i,j}|i=1,\cdots,\ N,j=0,1,\cdots,m_i\}$有定义，且$\lim\limits_{k\to\infty} x^k=x(1)$.若取$m_j\equiv 1$，且$t_i=\dfrac{i}{N}$，$H$取为（9.9.6）时，可得一个大范围收敛的Newton程序：

9.9.10
$$\begin{cases} x^{k+1}=x^k-F'(x^k)^{-1}[F(x^k)-(1-\dfrac{k}{N})F(x^0)] \\ \qquad\qquad\qquad\qquad (k=0,1,\cdots,N-1) \\ x^{k+1}=x^k-F'(x^k)^{-1}F(x^k) \qquad (k=N,N+1,\cdots) \end{cases}$$

9.9.11 **定理** 若$F:D\subset \boldsymbol{R}^n\to\boldsymbol{R}^n$有连续导数$F'(x)$，且$F'(x)$非奇并满足

$\|F'(x)^{-1}\|\leqslant\beta<+\infty,\quad \forall x\in D$

再假定在x^0的邻域$S=S\{x|\ \|x-x^0\|\leqslant\beta\ \|F(x^0)\|\ \}\subset \mathrm{int}D$存在$\gamma>0$，使

$\|F'(x)-F'(y)\|\leqslant\gamma\|x-y\|,\quad \forall x,y\in S$

则以x^0为初始值，存在正整数$N_0\geqslant 2\beta^2\gamma\|F(x^0)\|$，使当$N\geqslant N_0$时，数值延拓过程（9.9.10）收敛到方程$F(x)=0$在$S$中的唯一解$x^*=x(1)$，且具有平方敛速.

数值延拓法（9.9.10）实际上就是大范围收敛的Newton法，公式第一式是利用数值延拓法求出$x(1)$的一个足够好的近似x^N，使它进入Newton收敛域，从而保证Newton法收敛.

9.9.2 参数微分法

参数微分法是求同伦方程（9.9.2）的另一种数值方法。若满足（9.9.3）的映射 $x:[0, 1]\to D$ 可微，且 H 对 x 及 t 有连续偏导数 H_x 及 H_t，定义

$$\phi(t)=H(x(t), t), \quad \forall t\in[0,1]$$

则 ϕ 在 $[0, 1]$ 上连续可微，且

$$\phi'(t)=H_x(x(t),t)x'(t)+H_t(x(t),t), \quad \forall t\in[0,1]$$

因 $x=x(t)$ 满足（9.9.3），故有 $\phi'(t)=0$，于是 $x(t)$ 满足微分方程

9.9.12 $$H_x(x(t),t)x'(t)=-H_t(x(t),t), \quad \forall t\in[0,1]$$

反之，若 $x:[0, 1]\to D$ 是微分方程（9.9.12）的解，并满足 $H(x(0), 0)=0$，则由中值定理

$$\| H(x(t), t)\| = \|\phi(t)-\phi(0)\| \leqslant \sup_{0\leqslant s\leqslant t}\|\phi'(s)\| =0$$

有

$$H(x(t), t)=0$$

即微分方程（9.9.12）及初始条件 $H(x(0), 0)=0$ 的解 $x=x(t)$ 给出同伦方程（9.9.2）的一个解。若 H_x 非奇，$H(x^0, 0)=0$，则求（9.9.2）的解 $x=x(t)$ 等价于求微分方程初值问题

9.9.13 $$\begin{cases} x'(t)=-[H_x(x,t)]^{-1}H_t(x,t) \\ x(0)=x^0 \end{cases}$$

的解。于是，可用常微分方程初值问题的各种数值方法，求问题（9.9.13）在 $t_N=1$ 的数值解 x^N，作为 $x=x(1)$ 的近似解，它也就是方程（9.1.2）的解 x^* 的近似值。若同伦方程取为（9.9.6），对应于初值问题（9.9.13）可得

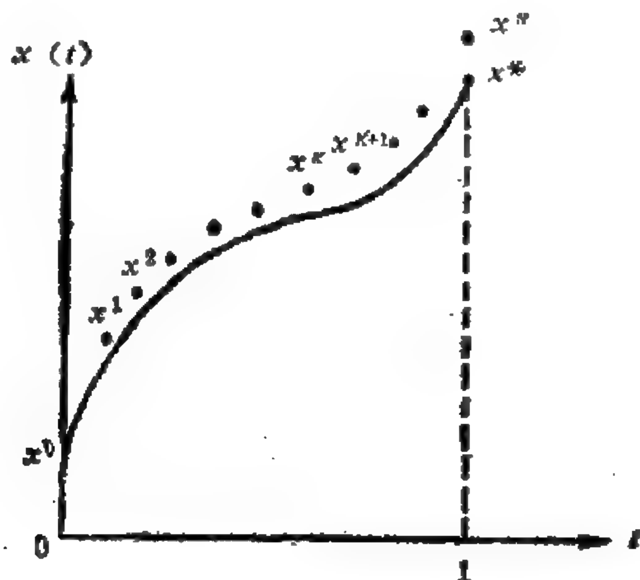
9.9.14 $$\begin{cases} x'(t)=-F'(x)^{-1}F(x^0), \quad \forall t\in[0,1] \\ x(0)=x^0 \end{cases}$$

若用 Euler 法求解，步长 $h=\frac{1}{N}$，则得

9.9.15　$x^{k+1}=x^k-hF'(x^k)^{-1}F(x^0)$　$(k=0,1,\cdots,N-1)$

实际上，当同伦方程有解时就是由(9.9.14)确定的一条解曲线 $x=x(t)$。当 h 充分小时，由（9.9.15）确定的 x^k $(k=0,1,\cdots,N)$ 应当近似这条曲线，x^N 可望与 $x^*=x(1)$ 充分靠近，且由 x^N 出发，用 Newton法迭代将收敛于 x^*。图(9.9.16）表示连接 x^0 与 x^* 的解曲线 $x=x(t)$ 及相应数值解 x^k。

9.9.16　图



当 $F: R^n \to R^n$，在 R^n 上二次连续可微并且$\|F'(x)^{-1}\| \leqslant \beta$时，对任何 $x^0 \in R^n$，存在一个 $N_0 \geqslant 1$，使得 $N \geqslant N_0$，则由（9.9.15）求得的 x^N 为初始近似，Newton迭代 $x^{k+1}=x^k-F'(x^k)^{-1}F(x^k)$　$(k=0,1,\cdots)$生成序列收敛于 $F(x)=0$ 的唯一解。

实际使用参数微分法时，还可用精度较高的数值方法求初值问题（9.9.14）的解。例如下列中点求积公式：

$$
9.9.17 \quad \begin{cases} x^1 = x^0 - \dfrac{1}{N} [F'(x^0)]^{-1} F(x^0) \\ x^{k+\frac{1}{2}} = x^k + \dfrac{1}{2} (x^k - x^{k-1}) \\ x^{k+1} = x^k - \dfrac{1}{N} [F'(x^{k+\frac{1}{2}})]^{-1} F(x^0) \ (K=1, \cdots, N-1) \end{cases}
$$

具有二阶精度，其计算工作量与 Euler 法相当，可用于实际计算。

9.9.18 **例** 用参数微分法求下列方程

$$
F(x) = \begin{pmatrix} f_1 \\ f_2 \end{pmatrix} = \begin{pmatrix} x_1^2 - x_2 + 1 \\ x_1 - \cos(\frac{\pi}{2} x_2) \end{pmatrix} = 0
$$

的近似解，给定初始近似 $x^0 = (1, 0)^T$.

解 此方程由给定x^0出发，直接用Newton法（9.3.1）迭代不收敛；若用 Newton下山法（9.3.17），要迭代107次才收敛到 $x^* = (0, 1)^T$；若用公式（9.9.17）。取 $N=8$可求得$x^N = (0.095552027, 0.97843313)^T$，从 x^N 出发用Newton迭代（9.3.1）做三次迭代，得

$x^* \approx (0.000000049, 0.999999963)^T$.

9.10

9.10 单纯形算法

单纯形算法(Simplicial Algorithm)是60年代后期出现的计算不动点和求非线性方程组零点的新算法，是拓扑学与计算数学结合的产物.算法的理论基础是Sperner(斯珀内)引理．1967年,H.Scarf(斯卡夫)首先提出了映射不动点的构造性逼近方法，体现了单纯形算法的思想，Cohen（柯亨）给出了Sperner 引理的构造性证明，随后Kuhn（库恩)、Eaves（伊维斯）等人又提出了几种新算法。这些方法不但在理论

上有重要意义，而且提供了一类具有大范围收敛的可行算法。

9.10.1 三角剖分与算法的基本思想

9.10.1 定义 设u^0，u^1，…，$u^m \in \boldsymbol{R}^n$在一般位置上，所有满足

9.10.2 $$x=\sum_{j=0}^{m}\lambda_j u^j,\quad \sum_{j=0}^{m}\lambda_j=1,\ \lambda_j\geqslant 0\quad (j=0,1,\cdots,m)$$

的点集合σ，称为一个**m维单纯形**(m-simplex)，记作

$\sigma=(u^0,u^1,\cdots,u^m)$

点u^0，u^1，…，u^m称为σ的**顶点**，记作

$\mathrm{Vert}(\sigma)=\{u^0,u^1,\cdots,u^m\}$

根据定义$m\leqslant n$，σ是一个m维凸多面体。$m=1$是直线段，$m=2$是平面上三角形，$m=3$是空间中的四面体。如果另有一个k维单纯形τ，满足

$\mathrm{Vert}(\tau)\leqslant \mathrm{Vert}(\sigma)$

则τ称为σ的一个**k维面**。由于$k\leqslant m$，所以，σ的零维面即σ的$m+1$个顶点，σ的一维面即多面体的棱，m维面是σ本身。σ的$m-1$维面称为σ的**真面**。如果τ的顶点中只比σ的顶点缺少u^j，记作$\tau=\mathrm{opp}(\sigma,u^j)$，称这个真面$\tau$为$\sigma$中$u^j$的**对面**。所有$m$个真面的并集称为$\sigma$的**边界**，记作

$$\partial(\sigma)=\bigcup_{j=0}^{m}\mathrm{opp}(\sigma,u^j)$$

9.10.3 定义 R^n中两个m维单纯形σ_1和σ_2称为**规则相联**的，如果$\sigma_1\neq\sigma_2$，且$\sigma_1\cap\sigma_2$是两者的一个$m-1$维公共面，如果有N个m维单纯形σ_1，…，σ_N两两之间规则相联，则其并集

$$D = \bigcup_{i=1}^{N} \sigma_i$$

称为 R^n 中的一个m**维复形**。σ_i 的集合

$$G = \{\sigma_i \mid i = 1, \cdots, N\}$$

称为 D 的一个**单纯形剖分**或**三角剖分**。

单纯形剖分 G 的格长记为 mesh(G)，

$$\text{mesh}(G) = \max_{\sigma \in G}\{\text{diam}(\sigma)\}$$

其中 $\text{diam}(\sigma) = \max\limits_{x, y \in \sigma} \|x - y\|$ 为单纯形 σ 的直径。

G 中所有单纯形的 k 维面集合称为 G 的 k **维骨架**，记作 G^k，于是 $G^m = G$，G^0 为 G 的顶点集合。关于 D 的单纯形剖分 G，有以下的直观性质：

1° 若 σ_1，$\sigma_2 \in G$ 且 $\sigma_1 \cap \sigma_2 \neq \phi$（空集），则 $\tau = \sigma_1 \cap \sigma_2 \in G^{m-1}$

2° 若 $\tau \in G^{m-1}$，那么，$\tau \subset \partial D$，τ 只是 G 的一个单纯形界面，或者 $\tau \not\subset \partial D$，$\tau$ 恰好是 G 中两个单纯形的公共界面；

3° 存在 G 的格长 mesh(G)任意小的剖分。

设 $F: D \subset R^n \to R^n$，G 为 D 上一个 n 维单纯形剖分，记 $N_0 = \{0, 1, \cdots, n\}$，则任何 $l: G^0 \to N_0$都可称为一个**整数标号**。例如，可由 F 定义 l，对任何 $u \in G^0$，定义

$$l(u) = \begin{cases} i, & \text{若} f_i(u) > 0, \text{且} f_j(u) \leqslant 0, \forall j < i \\ 0, & \text{若} f_j(u) \leqslant 0, j = 1, \cdots, n \end{cases} \tag{9.10.4}$$

这里 f_j 是 F 的第 j 个分量，即 $F = (f_1, \cdots, f_n)^T$。

G 中一个具有$\{0, 1, \cdots, n\}$标号的单纯形 σ，称为**全标号单纯形**。G 中具有$\{0, 1, \cdots, n-1\}$标号，而无 n 标号的 n 维单纯形称为**几乎全标号单纯形**。按（9.10.4）的标号定义，

有以下定理：

9.10.5 定理 设 $F:D\subset R^n\to R^n$，D 为有界闭凸集，若 F 在 D 上连续（即对任给的 $\varepsilon>0$，$\exists\delta>0$，对任何 x，$y\in D$，只要 $\|x-y\|\leqslant\delta$，就有 $\|F(x)-F(y)\|\leqslant\varepsilon$），若 D 的单纯形剖分 G 的格长 $\mathrm{mesh}(G)\leqslant\delta$，且 σ 为 G 中一个全标号单纯形，则对任何 $x\in\sigma$，有 $\|F(x)\|\leqslant\varepsilon$。

根据定理，在有界闭集 D 上求方程（9.1.2）的解，只要在 D 上建立一串 $\mathrm{mesh}(G_k)\to0$ 的单纯形剖分 $G_k(k=0,1,\cdots)$，在每个 G_k 上又有一个按（9.10.4）定义的全标号单纯形 σ_k，则 $\{\sigma_k\}$ 当 $k\to\infty$ 时有极限点 x^*，由定理（9.10.5）可知这个极限点 x^* 就是方程 $F(x)=0$ 的解。这就是单纯形算法的基本思想，算法包括三步：

1° 对 D 进行三角剖分 $G=\{\sigma_i|\ i=1,\cdots,N\}$；

2° 对 G 的顶点集合 G^0 进行标号；

3° 用一种有规则的搜索方法求 G 的全标号单纯形 σ。

9.10.6 例 用单纯形算法求方程

$$F(x,y)=\begin{bmatrix}x+y-1.5\\x^2-y^2-0.15\end{bmatrix}=0$$

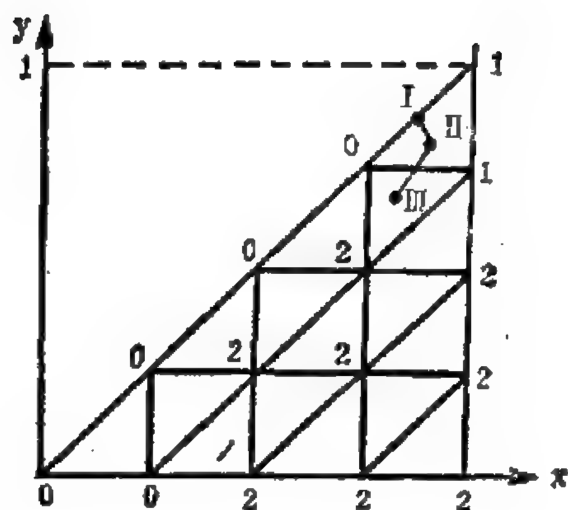
在(0,0)，(1,0)，(1,1)围成的区域 D 上的解。

解 第一步，先对域 D 进行三角剖分 G，如图（9.10.7）所示，这里用的是 $\frac{1}{4}K_1$ 剖分。

第二步，对各顶点标号，这里 $f_1=x+y-1.5$，$f_2=x^2-y^2-0.15$；按公式（9.10.4）计算得到 G 的各顶点标号，均标在图（9.10.7）上。

第三步，搜索全标号单纯形，这里用的方法是从 D 的边界一维全标号单形Ⅰ开始进入二维几乎全标号单形Ⅱ，再进入全标号单形Ⅲ，即为所求。若取Ⅲ的重心坐标 $x=0.875$，

9.10.7 图



$y=0.625$ 作为方程的近似解，此时 $f_1=0$，$f_2=0.225$．此题的精确解为 $x^*=0.8$，$y^*=0.7$．为提高精度可把剖分再缩小1/2，找到格长更小的全标号单纯形，其重心坐标为 $x=0.8125$，$y=0.6875$，更接近精确解．若精度不够，还可缩小剖分格长 mesh(G)，直到满足精度要求为止．

9.10.2 同伦算法

尽管单纯形算法具有只要求 F连续，且收敛是大范围的优点，但是，有两个严重弱点，一是当步长选定后，整个计算都要按此步长进行；二是搜索全标号单纯形必须从边上开始，这样步长取大了，精度很差；步长取小了，路径上节点很多，计算量太大，使得实际计算几乎不可能．因此，单纯形算法只能作为求初始近似的方法．针对单纯形算法的弱点，Merrill于 1972 年提出了**重启动**(Restart)法，Eaves提出了**同伦**（Homotopy）**算法**，又称**连续变形算法**（Continuous Deformation Algorithm)．这两种方法从理论分析和计算实践都表明是可行的，此后出现的新算法均以此为基础．

定义同伦 $H:D\times[0,1]\subset R^{n+1}\to R^n$ 为

$$
\text{9.10.8}\quad H(x,t)=\begin{cases}2tF(x)+(1-2t)F_0(x), & 0\leqslant t\leqslant 1/2\\ F(x), & 1/2\leqslant t\leqslant 1\end{cases}
$$

满足条件（9.9.1）。对 $D\times[0,1]$ 进行三角剖分 G，它具有以下性质：对 $D\times[1-2^{-k},1-2^{-k-1}]$ 的三角剖分 $G_k(k=0,1,\cdots)$，为 G 的子集，且当 $k\to\infty$，$\text{mesh}(G_k)\to 0$，$n=1$ 时，这种剖分如图（9.10.13）所示。

9.10.9 定义 设 $F:D\in \boldsymbol{R}^n\to\boldsymbol{R}^n$，对凸集 D 有三角剖分 G，$\sigma=\langle v^0,\cdots,v^n\rangle\in G$，对任何 $x\in\sigma$，若 $x=\sum_{i=0}^{n}\lambda_i v^i$，$\lambda_i\geqslant 0$ 且 $\sum_{i=0}^{n}\lambda_i=1$，则定义 $P:D\subset\boldsymbol{R}^n\to\boldsymbol{R}^n$ 为

$$
P(x)=\sum_{i=0}^{n}\lambda_i F(v_i)
$$

称为基于 G 的一个**分片线性逼近**（Piecewise linear Approximation）。$P(x)$ 可做为 F 的一个向量标号函数。

若 $P(x^*)=0$，$x^*\in D$，则称 x^* 为 F 在三角剖分 G 下的一个近似解。若 $x^*\in\sigma\in G$ 则 σ 称为一个 n 维**完全**（Complete）**单纯形**。

9.10.10 定理 设 $F:D\subset\boldsymbol{R}^n\to\boldsymbol{R}^n$ 连续可微，且 $F'(x)$ 在凸集 D 上Lipschitz连续，即 $\exists\gamma>0$，使

$\|F'(x)-F'(y)\|\leqslant\gamma\|x-y\|$，$\forall x,y\in D$

若 x^* 是 F 在三角剖分 G 下的近似解，$\text{mesh}(G)\leqslant\delta$，则有

$\|F(x)\|\leqslant\gamma\delta^2$

（这里 $\|\cdot\|$ 均为 l_∞ 范数）。

根据定理只要找到完全单纯形 σ，也就找到了 $F(x)=0$ 的近似解，它比全标号单纯形具有更高的精度，问题是如何找

到完全单纯形。当$F:D\subset \boldsymbol{R}^n\to\boldsymbol{R}^n$，$\sigma=\langle v^0, \cdots, v^n\rangle\in G$时，称$(n+1)$阶矩阵

$$L_\sigma=\begin{pmatrix}1 & 1 & \cdots & 1\\ F(v^0) & F(v^1) & \cdots & F(v^n)\end{pmatrix}$$

为σ的**标号矩阵**(Label Matrix)。

σ为n维完全单纯形的充分必要条件是

$L_\sigma w=e^1, w\geqslant 0, w\in\boldsymbol{R}^{n+1}, e^1=(1,0,\cdots,0)^T\in\boldsymbol{R}^{n+1}$

有解。

设同伦$H:D\times[0, 1]\subset\boldsymbol{R}^{n+1}\to\boldsymbol{R}^n$，对$D\times[1-2^{-k}, 1-2^{-k-1}]$的三角剖分$G_k$ $(k=0, 1, \cdots)$，若

$\tau=\langle v^0, \cdots, v^n, v^{n+1}\rangle\in G_k\subset G$

同样称$(n+1)\times(n+2)$阶矩阵

$$L_\tau=\begin{pmatrix}1 & 1 & \cdots & 1\\ H(v^0) & H(v^1) & \cdots & H(v^{n+1})\end{pmatrix}$$

为τ的标号矩阵。当

9.10.11 $\quad L_\tau w=e^1, w\geqslant 0, w\in\boldsymbol{R}^{n+2}, e^1=(1,0,\cdots,0)^T\in\boldsymbol{R}^{n+1}$

有解时，称τ为一个$(n+1)$维**几乎完全单纯形**。根据线性规划基本定理，当矩阵L_τ的秩为$(n+1)$时，若(9.10.11)有解，则它恰好有两个基本可行解。这就说明，G中一个$(n+1)$维几乎完全单纯形τ，当L_τ的秩为$n+1$时恰有两个n维面是完全单形，定义这样的两个n维单形是相邻的。另一方面，对不在$D\times\{1-2^{-k}\}$ $(k=0, 1, \cdots)$上的n维完全单形，它一定正好属于两个G中的$(n+1)$维几乎完全单形。于是，当x^0为$D\times\{0\}$上的一个n维完全单形内点时，由此开始形成一个唯一的完全单形轮回路径，实际上就是同伦方程$H(x, t)=0$，对$t\in[0, 1]$的解的轨迹，当$t\to 1$(即$k\to\infty$)，解$x(t)$就趋向$F(x)=0$的解。

设 $\sigma_0=(v^0, \cdots, v^n)$是三角剖分 G_0 包含$(x^0, 0)^T$的一个 n 维面，σ_0 是 n 维完全单纯形，于是同伦算法由此开始，其算法步骤如下：

1° 由 σ_0 求 $\tau_0=\langle v^0, \cdots, v^n, v^{n+1}\rangle\in G_0$，$0\to p$。

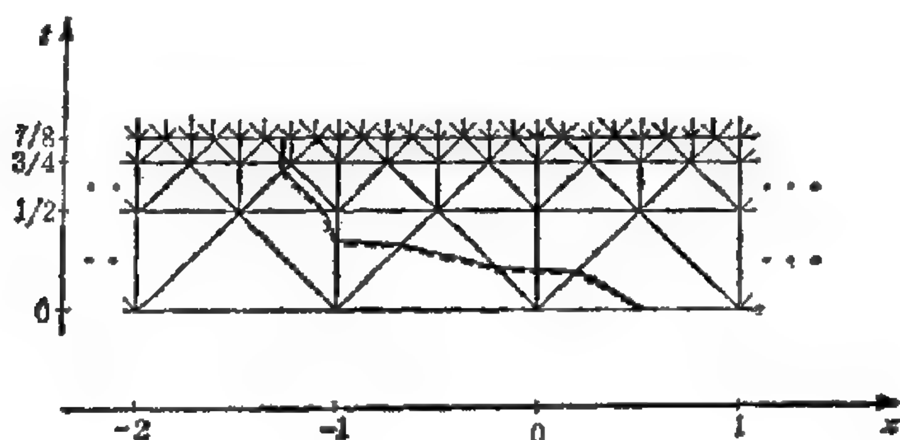
2° 设 σ_p 是 τ_p 的完全单形 n 维面，由 τ_p 求另一个完全单形 n 维面 σ_{p+1}。

3° 若 $\sigma_{p+1}\in D\times\{1-2^{-k}\}$，且$2^{-k}\leqslant\varepsilon$（给定精度），则转 4°；否则，由 σ_{p+1} 求 $\tau_{p+1}\neq\tau_p$（τ_{p+1} 与 τ_p 规则相联），$p+1\to p$ 并转2°。

4° $\sigma_{p+1}=\langle u^0, \cdots, u^n\rangle=\langle(x^0, 1-2^{-k})^T, \cdots, (x^n, 1-2^{-k})^T\rangle$是一个 n 维完全单纯形，存在权 $w_0, \cdots, w_n$ 使 $u^*=\sum_{i=0}^{n}w_iu^i$ 是 $H(x, 1-2^{-k})$的一个零点，而 $x^*=\sum_{i=0}^{n}w_ix^i$ 是 $f(x)$的一个零点。

9.10.12 例 令 $n=1$，$F(x)=x^3+2$，$F_0(x)=x-1/2$，$D\times[0, 1]$的三角剖分 G 及$H^{-1}(0)$的轨迹如图（9.10.13)所示。

9.10.13 图



9.11 区间迭代法

求解非线性方程组

9.11.1 $f(x)=0 \quad (f:D\subset \boldsymbol{R}^n\to \boldsymbol{R}^n)$

的区间迭代法，首先是由 Moore（莫尔）于 1966 年提出的，它以区间为变量，将区间映到区间，Moore 给出的区间 Newton 算子是

9.11.2 $N(X)=x-F'(X)^{-1}f(x),\ \forall x\in X$

这里 $X\subset D\subset \boldsymbol{R}^n$ 为 $\boldsymbol{R}^n$ 中的区间向量，$X=(X_1,\ \cdots,\ X_n)^T$ 是一个 n 维长方体，X_i 都是区间. $F'(X)$ 是 $f'(x)$ 的包含区间扩展,即对任何 $x\in X$,有 $f'(x)\in F'(X)$. $F'(X)^{-1}$ 就是一个区间矩阵的逆. 因此，(9.11.2) 给出的算子 $N(X)$ 仍然是一个 n 维区间向量，并有以下性质：

1° 若 $x^*\in X$ 是 (9.11.1) 的解，则 $x^*\in N(X)$；

2° 若 $X\cap N(X)=\phi$(空集)，则 (9.11.1) 在 X 中无解；

3° 若 $N(X)\subset X$ 则 (9.11.1) 在 X 中有解 x^*，且 $x^*\in N(x)$；若 $W(N(X))<W(X)$($W(X)$ 表示区间 X 的宽度,当 $X=(X_1,\ \cdots,X_n)^T$，宽度 $W(X)=(W(_1X),\cdots,W(X_n))^T$)，则 (9.11.1) 在 X 中的解 x^* 是唯一的.

根据 $N(X)$ 的性质，可构造区间 Newton迭代

9.11.3 $X^{k+1}=X^k\cap N(X^k) \quad (k=0,1,\cdots)$

若对某个 k 有 $N(X^k)\subset X^k$，且 $W(N(X^k))<W(X^k)$，则 $\lim\limits_{k\to\infty}X^k=x^*$ 为方程 (9.11.1) 的解；若 $X^k\cap N(X^k)=\phi$，则 (9.11.1) 在 X^0 中无解. 这表明区间 Newton 法每步迭代都可检验方程 (9.11.1) 解的存在唯一性，这是点迭代法

所不能解决的，也是区间迭代法的主要优点．但是，求$N(X)$要计算区间矩阵$F'(X)$的逆，要按区间运算规则求$f'(x)$的区间扩展$F'(X)$，再求逆，因此，解区间线性方程组的工作量很大，故很不实用．为减少计算工作量，避免求逆，Krawczyk（克拉夫楚克）在算子$N(X)$基础上引进新的区间算子

9.11.4 $K(y,X)=y-Yf(y)+[I-YF'(X)](X-y)$

其中$y\in X$是X中任一点，Y为非奇的n阶矩阵，即$Y\in R^{n\times n}$，I为n阶单位矩阵．$F'(X)$仍为$f'(x)$的区间扩展，称$K(y,X)$为Krawczyk算子．区间向量$X=[\underline{x},\overline{x}]=\{x\in R^n|\underline{x}\leqslant x\leqslant\overline{x}\}$是$R^n$中的紧凸集，由不动点定理可知，若$\overline{N}(x)=x-Yf(x)$映区间$X$于自身，即$\overline{N}(X)\subset X$，则存在$x^*\in X$为$\overline{N}(x)$的不动点，于是$f(x^*)=0$，而对任何$x$，$y\in X$，有

$$\overline{N}(x)-\overline{N}(y)=\overline{N}'(\xi)(x-y),\forall\xi\in X,\overline{N}'(x)=I-Yf'(x)$$

故

$$\overline{N}(x)\in K(y,X)=\overline{N}(y)+\overline{N}'(X)\ (X-y)$$

根据Krawczyk算子的这一性质，可以得到检验方程(9.11.1)解存在唯一性的定理．

9.11.5 **定理** 设$f:X=[\underline{x},\overline{x}]\subset \boldsymbol{R}^n\to\boldsymbol{R}^n$在$X$中连续可微，且$f$，$f'$有包含区间扩展$F$及$F'$，$K(y,X)$是由(9.11.4)定义的Krawczyk算子，则

1° 方程(9.11.1)在X中的解都在$K(y,X)$中；

2° 若$X\cap K(y,X)=\phi$，则(9.11.1)在X中无解，若$K(y,X)\subset X$，则方程(9.11.1)在X中至少有一个解；

3° 若$K(y,X)\subset X$，且$W(K(y,X))<W(X)$，则方程(9.11.1)在X中有唯一解x^*，且以$x^0\in X$为初始近似的迭代序列

$$x^{(k+1)}=x^{(k)}-Yf(x^{(k)})\quad(k=0,1,\cdots)$$

收敛到x^*，且

$$|X^{(k)}-x^*|\leqslant\beta^k W(X)$$

其中

$$\beta=\max_{1\leqslant i\leqslant n}\left\{W(K(y,X)_i)\Big/W(X_i)\right\}<1$$

此定理也称为 Moore 检验，它可通过下列区间迭代法验检解的存在唯一性。

9.11.6 算法

取 $X^0=X=[\underline{x},\overline{x}]$，$y^0=m(X)$（$m(X)=\dfrac{\underline{x}+\overline{x}}{2}$称为区间$X$的中点）。对 $k=0$，1，…，计算

$$K(y^k,X^k)=y^k-Y_kf(y^k)+[I-Y_kF'(X^k)](X^k-y^k)$$

其中 $y^k=m(X^k)$，$Y_k=[m(F'(X^k))]^{-1}$。若 $X^k\cap k(y^k,X^k)=\phi$，则停止计算，方程在 x 中无解；否则令

$$X^{k+1}=X^k\cap K(y^k,X^k)$$

若$|W(X^{k+1})|\leqslant\varepsilon$（给定精度），则停止计算，并取 $m(X^{k+1})$ 作为 x^* 近似；否则 $k+1\to k$ 重新计算。

只要存在某个l，使$K(y^l,X^l)\subset X^l$，且$W(K(y^l,X^l))<W(X^l)$成立，则

$$\lim_{k\to\infty}X^k=x^*$$

算法（9.11.6）以及Moore 检验（9.11.5）体现了区间迭代法的特点，计算量也比区间 Newton 法（9.11.3）大为减少，它可直接用来判断方程组（9.11.1）在区域$X=[\underline{x},\overline{x}]$中是否有解以及解是否唯一，如解存在唯一还可得到近似解的误差估计。

1980年，Nickel(尼克尔)给出一种球形 Newton 法，它是区间迭代法的另一个杰出成就。在 $\boldsymbol{R}^n$ 中的 n 维球 $A=\{x\in\boldsymbol{R}^n\mid\|x-a\|\leqslant r\}$可表示为$A=\langle a,r\rangle$，$a\in\boldsymbol{R}^n$ 为中心，记作 $\mathrm{mid}(A)=a$；r 为半径，记作 $\mathrm{rad}(A)=r$。对 $x\in\boldsymbol{R}^n$，

定义正则球算子

$Lx=\langle Ax,\ \lambda\|Ax\|\rangle$

其中 A 为 n 阶非奇矩阵，$\lambda\in(0,\ 1)$表示$L=\langle\langle A,\ \lambda\rangle\rangle$.设$f:D\subset R^n\to R^n$，对每个形如

$B=\langle a,r\rangle\cap D,\ a\in D$

的集合，如果存在非奇矩阵 $A=A(B)$和正实数$\lambda\in(0,\ 1)$，有

$\|x-y-A(f(x)-f(y))\|\leqslant\lambda\|A(f(x)-f(y))\|$

$\forall x,y\in B$成立，则所有满足这些要求的函数f统称为函数类F。若$f\in F$，可定义球形 Newton算子

9.11.7 $Nx=x-Lf(x)=\langle x-Af(x),\lambda\|Af(x)\|\rangle,\ x\in B$

显然，Nx是一个球，且

$\mathrm{mid}\ (Nx)\ =x-Af\ (x)\ ,\ \mathrm{rad}\ (Nx)=\lambda\|Af\ (x)\|$

对球形 Newton算子 N，也有类似于区间Newton算子的性质，即

1°若 $x^*\in B$，则 $f\ (x^*)\ =0$ 的充分必要条件是 $x^*=Nx^*$，求方程(9.11.1)的解等价于求 N 在 B 上的不动点；

2°若 $x\in B$ 且 $B\cap Nx=\phi$，则方程 (9.11.1) 在B中无解；

3° 若对 $\forall x\in B$ 有 $Nx\in B$，则 (9.11.1) 在 B 中有解 x^*存在；

4°若$x^*\in B$，$f\ (x^*)\ =0$，则 $x^*\in Nx$.

利用球形 Newton算子 N 可构造球形 Newton法。

9.11.8 **算法**（球形 Newton法）

初始步：取 $X^0=D=\langle x^0,r^0\rangle$计算 Nx^0。

1° 若 $X^0\cap Nx^0=\phi$，则算法停止。

2° 若 $X^0\cap Nx^0\neq\phi$，则定义交球（交球是指包含两球交的最小球）

$X^{(1)}=\langle X^0\cap Nx^0\rangle$

计算步： 假定 X^k $(K\geqslant 1)$ 已定义，且$x=\mathrm{mid}X^k\in X^0$，已计算出了球 Nx^k。

1° 若 $X^k\cap Nx^k=\phi$，则算法停止。

2° 若 $X^k\cap Nx^k\neq\phi$，则新的球定义为

$\overline{X}^{k+1}=\langle X^k\cap NX^k\rangle$

若$\overline{X}^{k+1}\cap X^0=\phi$，则算法停止。若$X^{k+1}\cap\overline{X}^0\neq\phi$，且$X^{k+1}=\mathrm{mid}\,\overline{X}^{k+1}$。则当$\overline{x}^{k+1}\in X^0$ 定义 $X^{k+1}=\overline{X}^{k+1}$；否则定义 $X^{k+1}=\langle X^0\cap\overline{X}^{k+1}\rangle$。这样定义的球形 Newton法是全局收敛的。球形 Newton迭代同样具有区间迭代的特点，但计算交球仍然较复杂。

10

第10章　常微分方程初值问题的数值方法

10.1

10.1　引　言

10.1.1　常微分方程的初值问题

一阶常微分方程初值问题的一般形式为

10.1.1　$\frac{dy}{dx}=f(x,y)$

10.1.2　$y(x_0)=y_0$

其中（10.1.1）为微分方程，式中 f 是已知的函数；（10.1.2）为初始条件，式中 y_0 是已知的值。

10.1.3　**定理**　设 f 为x，y的连续函数，则初值问题（10.1.1），（10.1.2）在包含 x_0 的区间上有解 $y(x)$的充分必要条件为 $y(x)$ 满足

10.1.4　$y(x)=y_0+\int_{x_0}^{x}f(t,y(t))dt$

10.1.5　**定义/Lipschitz（李普希兹）条件/Lipschitz Condition**　若存在常数 $L>0$，使 $f(x,y)$在区域 D 上对所有(x,y_1)，$(x,y_2)\in D$，有

10.1.6　$|f(x,y_1)-f(x,y_2)|\leqslant L|y_1-y_2|$

则称 f 在 D 上对y满足 Lipschitz **条件**，常数L称为Lipschitz **常数**。

10.1.7　**定理**　若f在$D=\{(x,y)|x_0\leqslant x\leqslant b,-\infty<y<+\infty\}$上连续，且对 y 满足 Lipschitz 条件，则初值问题（10.1.1），（10.1.2）对任意给定的 y_0，在$x_0\leqslant x\leqslant b$上存在唯一的解

y，其中 y 在 $x_0 \leqslant x \leqslant b$ 上连续可微。

一阶常微分方程组初值问题的一般形式为

10.1.8
$$\frac{dy_1}{dx}=f_1(x,y_1,y_2,\cdots,y_n),\qquad y_1(x_0)=y_{10}$$
$$\frac{dy_2}{dx}=f_2(x,y_1,y_2,\cdots,y_n),\qquad y_2(x_0)=y_{20}$$
$$\vdots\qquad\qquad\qquad\vdots$$
$$\frac{dy_n}{dx}=f_n(x,y_1,y_2,\cdots,y_n),\qquad y_n(x_0)=y_{n0}$$

高阶常微分方程初值问题的一般形式为

10.1.9
$$\frac{d^ny}{dx^n}=f\left(x,y,\frac{dy}{dx},\cdots,\frac{d^{n-1}y}{dx^{n-1}}\right)$$

10.1.10
$$y(x_0)=y_0,\ \frac{dy(x_0)}{dx}=y_0^{(1)},\cdots,\ \frac{d^{n-1}y(x_0)}{dx^{n-1}}=y_0^{(n-1)}$$

其中（10.1.9）是 n 阶的微分方程，(10.1.10)是初始条件。

如果设

$$y_1=y,\ y_2=\frac{dy}{dx},\ \cdots,\ y_n=\frac{d^{n-1}y}{dx^{n-1}}$$

则（10.1.9），（10.1.10）可化为方程组的形式：

$$\frac{dy_1}{dx}=y_2,\qquad y_1(x_0)=y_0$$
$$\vdots\qquad\qquad\vdots$$
$$\frac{dy_{n-1}}{dx}=y_n,\qquad y_{n-1}(x_0)=y_0^{(n-2)}$$
$$\frac{dy_n}{dx}=f(x,y_1,y_2,\cdots,y_n),\qquad y_n(x_0)=y_0^{(n-1)}$$

10.1.2 数值离散方法

考虑初值问题（10.1.1）和（10.1.2）在区间$x_0 \leqslant x \leqslant b$上的离散解，引入点列

$$x_n = x_{n-1} + h_n \qquad (n=1,2,\cdots)$$

其中$h_n > 0$。称x_n为**节点**，h_n为**步长**。通常考虑h_n为常数的情形，即$h_n = h, n=1,2,\cdots$，称为**等步长**的情形，这时

10.1.11 $$x_n = x_0 + nh \qquad (n=0,1,\cdots)$$

取h使$\frac{b-x_0}{h}$为正整数，求$x_0 \leqslant x \leqslant b$上的离散解，就是在点列$\left\{x_n \mid n=0,1,\cdots,\frac{b-x_0}{h}\right\}$上求初值问题解的近似值。记（10.1.1），（10.1.2）的准确解y在x_n点之值为$y(x_n)$，而近似值记为y_n，并记$f_n = f(x_n, y_n)$。

10.1.12 **定义/线性多步法/Linear Multistep Method** 若确定序列$\{y_n\}$的一个计算方法是由y_{n+j}，$f_{n+j}(j=0,1,\cdots,k)$的线性关系所组成，即

10.1.13 $$\alpha_0 y_n + \alpha_1 y_{n+1} + \cdots + \alpha_k y_{n+k} = h(\beta_0 f_n + \beta_1 f_{n+1} + \cdots + \beta_k f_{n+k})$$

则称此计算方法为步数为k的**线性多步法**，其中，设$\alpha_k \neq 0$，α_0、β_0两者不能全为0。

因为（10.1.13）全式可乘一常数因子，可规定$\alpha_k = 1$，此时（10.1.13）可写成

10.1.14 $$y_{n+k} = -\alpha_0 y_n - \alpha_1 y_{n+1} - \cdots - \alpha_{k-1} y_{n+k-1} + h(\beta_0 f_n + \beta_1 f_{n+1} + \cdots + \beta_k f_{n+k})$$

如果已知$y_n, y_{n+1}, \cdots, y_{n+k-1}$之值，就可通过（10.1.14）计算$y_{n+k}$。

10.1.15 **定义/单步法/One-Step Method** 从y_n之值计算y_{n+1}的方法，称为**单步法**。

在（10.1.13）或（10.1.14）中，若$k=1$，就是线性的单步法，对于非线性的单步法，可以写成

10.1.16 $y_{n+1}=y_n+h\phi(x_n,y_n,x_{n+1},y_{n+1},h)$

由（10.1.2），根据初值 y_0 即可由（10.1.16）逐步计算 $y_1,y_2,\cdots$. 对于线性多步法，则要另外计算初始值 $y_0,y_1,\cdots,y_{k-1}$，然后用（10.1.14）逐步计算 $y_k,y_{k+1},\cdots$.

10.1.17 **定义/显式方法/Explicit Method** 在(10.1.14)中，若 $\beta_k=0$，则称此计算方法为**显式的线性多步法**。在(10.1.16)中，若函数 ϕ 不显含 y_{n+1}，则此计算方法为**显式的单步法**.

显式单步法一般可表示为

10.1.18 $y_{n+1}=y_n+h\phi(x_n,y_n,h)$

10.1.19 **定义/隐式方法/Implicit Method** 在(10.1.14)中，若 $\beta_k\neq0$，则称此计算方法为**隐式的线性多步法**。在(10.1.16)中，若函数 ϕ 显含 y_{n+1}，则称此计算方法是**隐式的单步法**.

对于显式的线性多步法，y_{n+k} 可由 $y_{n+j},f_{n+j}(j=0,1,\cdots,k-1)$ 直接计算出来。对于显式的单步法(10.1.18)，y_{n+1} 也可由 y_n 直接计算出来。对于隐式的线性多步法，可将（10.1.14）改写为

10.1.20 $y_{n+k}=h\beta_k f(x_{n+k},y_{n+k})+g$

其中

$$g=\sum_{j=0}^{k-1}(-\alpha_j y_{n+j}+h\beta_j f_{n+j})$$

是可以在计算 y_{n+k} 之前预先计算好的。如果 f 是非线性的函数，（10.1.20）就是 y_{n+k} 的非线性方程，一般可用如下的迭代方法求解：

10.1.21
$$\begin{cases} y_{n+k}^{[0]} & \text{任意} \\ y_{n+k}^{[s+1]}=h\beta_k f(x_{n+k},y_{n+k}^{[s]})+g & (s=0,1,2\cdots,) \end{cases}$$

到 $|y_{n+k}^{[s+1]}-y_{n+k}^{[s]}|<\varepsilon$ 时迭代结束，其中 ε 是预先给定的误差限.

10.1.22 **定理** 若 f 对 y 满足 Lipschitz 条件，L 为其Lipschitz

常数（参见10.1.5，10.1.6），并设 $h<1/(L|\beta_k|)$，则由（10.1.21）得到的序列$\{y_{n+k}^{[s]}\}$收敛到(10.1.20)的解y_{n+k}.

10.2

10.2 显式单步法的一般概念

如（10.1.18）所示，解初值问题(10.1.1),(10.1.2)的显示单步法为

10.2.1 $y_{n+1}=y_n+h\phi(x_n,y_n,h)$

10.2.2 **定义/阶/Order** 设$y(x)$是初值问题(10.1.1),(10.1.2)的准确解，(10.2.1) 是解此初值问题的一种显式单步法，则满足

$$y(x+h)-y(x)-h\phi(x,y(x),h)=O(h^{p+1})$$

的最大整数 p，称为（10.2.1）的阶.

10.2.3 **定义/相容性/Consistency** 若

$$\phi(x,y,0)=f(x,y)$$

则称（10.2.1）与初值问题(10.1.1)，(10.1.2) 是相容的.

若（10.2.1）与初值问题(10.1.1)，(10.1.2)相容，利用 Taylor 公式在 $h=0$ 展开，有

$$y(x+h)-y(x)-h\phi(x,\ y(x),\ h)$$

$$=hy'(x)-h\phi(x,y(x),\ 0)+O(h^2)=O(h^2)$$

所以，与初值问题相容的方法至少是一阶的方法.

10.2.4 **例/Euler（欧拉）方法/Euler Method**

$$y_{n+1}=y_n+hf(x_n,y_n)\qquad(n=0,1,2\cdots)$$

这里，$\phi(x,y,h)=f(x,y)$，所以 Euler方法与初值问题显然是相容的。因为

$$y(x+h)-y(x)-h\phi(x,y(x),h)$$

$$=y(x+h)-y(x)-hf(x,y)$$

$$=y(x+h)-y(x)-hy'(x)=O(h^2)$$

所以 Euler 方法是一阶的显式单步法.

10.2.5 **定义/收敛性/Convergence** 初值问题（10.1.1），

(10.1.2) 的一种数值方法对任意固定的 $x_n=x_0+nh$，当 $h\to 0$（同时$n\to\infty$）时有 $y_n\to y(x_n)$，称该方法是**收敛的**.

在实际计算中使用的方法，应该是相容的和收敛的.

10.2.6 定义/整体截断误差/Global Truncation Error

$e_n=y(x_n)-y_n$ 称为一种数值方法在 x_n 点的**整体截断误差**.

定义(10.2.5)，(10.2.6)不只是对显式单步法的，对其它数值方法亦适用.对于显式单步法(10.2.1)，有如下定理：

10.2.7 定理 设在区域$D=\{(x,y,h)|\ x\in[x_0,b],\ y\in(-\infty,\infty),h\in[0,h_0]\}$ $(h_0>0)$ 中，函数 ϕ 满足：

1° 作为三个自变量的函数，在D上连续；

2° 对 y 满足 Lipschitz 条件；

则显式单步法 (10.2.1) 收敛的充分必要条件是它与初值问题相容；而且，若 (10.2.1) 是 p阶的方法，则整体截断误差

$$y(x_n)-y_n=O(h^p)$$

10.2.8 定义/局部截断误差/Local Truncation Error

$$T_{n+1}=y(x_{n+1})-y(x_n)-h\phi(x_n,y(x_n),h)$$

称为 (10.2.1) 在x_{n+1}的**局部截断误差**，其中$y(x)$是初值问题(10.1.1)，(10.1.2)的准确解.

如果假设以上各步没有误差，即 $y(x_n)=y_n$ 准确成立，则有$T_{n+1}=y(x_{n+1})-y_{n+1}$，这就是形容词"局部"的意义.显然，$T_{n+1}$ 与 e_{n+1} 的意义是不相同的.

如果将 (10.2.8) 中 T_{n+1} 的表达式按 h 的幂展开，则对于p 阶的方法，有

$$T_{n+1}=O(h^{p+1})$$

10.2.9 定义/主局部截断误差/Principal Local Truncation Error 若 (10.2.1) 是 p 阶的方法，其局部截断误差可写成

$$T_{n+1}=\psi(x_n,y(x_n))h^{p+1}+O(h^{p+2})$$

则称$\psi(x_n, y(x_n))h^{p+1}$为（10.2.1）在x_{n+1}的**主局部截断误差**。

常用的显式单步方法，有Euler方法，Runge-Kutta方法等。

10.3

10.3 Euler方法

10.3.1 Euler（欧拉）方法

（10.2.4）已经给出计算初值问题(10.1.1),(10.1.2)的一种显式方法——Euler方法，即

10.3.1 $y_{n+1}=y_n+hf(x_n, y_n)\quad(n=0,1,2,\cdots)$

Euler方法可以看成在（10.1.1）中，$y'(x)$用$\dfrac{y(x+h)-y(x)}{h}$近似代替而得到的方法，所以它是一种最简单的显式方法。Euler方法又可看成在式（10.1.13）中，令$k=1$（即单步方法），系数取为$\alpha_0=-1$，$\alpha_1=1, \beta_0=1$，$\beta_1=0$的情况，所以是显式的线性单步法。

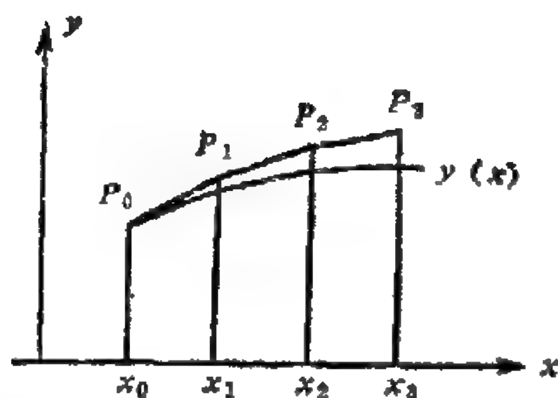
如果把初值问题看成类似（10.1.4）的形式，有

10.3.2 $$y(x_{n+1})=y(x_n)+\int_{x_n}^{x_{n+1}} f(t, y(t))dt$$

设$y_n=y(x_n)$，上式右端的积分式若用左矩形数值积分公式近似计算，可得到$y(x_{n+1})$的近似值y_{n+1}即（10.3.1），所以Euler方法也可以认为是由数值积分公式构造出来的。

10.3.3 Euler方法的几何解释

设初值问题（10.1.1），（10.1.2)的准确解为$y(x)$，其曲线过P_0（x_0, y_0）点。Euler方法可以看成从P_0出发，按f（x_0, y_0）为斜率的方向作一直线段，向前推进到$x=x_1$上一点P_1（x_1, y_1）；再从P_1出发，按f（x_1, y_1）为斜率的方向，推进到$x=x_2$上一点P_2（x_2, y_2），依此类推。这样得



到一条折线 $\overline{P_0P_1P_2\cdots\cdots}$，近似于曲线 $y(x)$。

Euler 方法是一阶的方法，它的局部截断误差为

$$\frac{h^2}{2}y''(x_n)+O(h^3).$$

10.3.4 例 用 Euler 方法近似求解初值问题

$$\frac{dy}{dx}=-y+x+1,\quad 0\leqslant x\leqslant 1$$

x_n	y_n	$y(x_n)$	$\lvert y(x_n)-y_n\rvert$
0	1.000000	1.000000	0
0.1	1.000000	1.004837	0.004837
0.2	1.010000	1.018731	0.008731
0.3	1.029000	1.040818	0.011818
0.4	1.056100	1.070320	0.014220
0.5	1.090490	1.106531	0.016041
0.6	1.131441	1.148812	0.017371
0.7	1.178297	1.196585	0.018288
0.8	1.230467	1.249323	0.018862
0.9	1.287420	1.306570	0.019150
1.0	1.348678	1.367879	0.019201

$y(0)=1$

取步长 $h=0.1$。

解 利用（10.3.1），其中 $x_n=nh=0.1n$，

$f(x_n,y_n)=-y_n+x_n+1$

$y_0=1$

$$
\begin{aligned}
y_{n+1}&=y_n+h(-y_n+x_n+1)\\
&=(1-h)y_n+hx_n+h\\
&=0.9y_n+0.1x_n+0.1 \quad (n=0,1,\cdots,9)
\end{aligned}
$$

计算结果与准确解 $y(x)=x+e^{-x}$ 的比较如上表。

10.3.2 隐式 Euler 方法和梯形方法

如果将（10.3.2）式中右端的积分式用右矩形数值积分公式近似计算，并设 $y_n=y(x_n)$，就得到

10.3.5 隐式 Euler 方法/Implicit Euler Method

$$y_{n+1}=y_n+hf(x_{n+1},y_{n+1})$$

（10.3.5）相当于在（10.1.13）中，令 $k=1$，系数取成 $\alpha_0=-1$，$\alpha_1=1$，$\beta_0=0$，$\beta_1=1$，所以它是隐式的单步法。若设 $y_n=y(x_n)$，则

$$y(x_{n+1})-y_{n+1}=-\frac{h^2}{2}y''(x_n)+O(h^3)$$

为了得到比 Euler 方法更为精确的解，可以用数值积分方法中的梯形方法来近似（10.3.2）右端的积分，得到解初始问题（10.1.1），（10.1.2）的梯形方法。

10.3.6 梯形方法/Trapezoidal Method

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1})]$$

（10.3.6）相当于在（10.1.13）中，令 $k=1$，系数取成 $\alpha_0=-1$，$\alpha_1=1$，$\beta_0=\beta_1=\frac{1}{2}$，所以它是一个单步的隐式方

法。为了解出y_{n+1}，可采用（10.1.21）的迭代方法，其中的迭代初值$y_{n+1}^{[0]}$就取为 Euler 方法计算所得之值，即

10.3.7
$$\begin{cases} y_{n+1}^{[0]} = y_n + hf(x_n, y_n) \\ y_{n+1}^{[s+1]} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_{n+1}^{[s]})] \end{cases}$$

$$(s=0,1,2,\cdots)$$

到$|y_{n+1}^{[s+1]} - y_{n+1}^{[s]}| < \varepsilon$时迭代结束，$\varepsilon$为预先给定的误差限。

对于梯形方法，若设$y(x_n) = y_n$，则

$$y(x_{n+1}) - y_{n+1} = -\frac{h^3}{12}y'''(x_n) + O(h^4)$$

10.3.3 改进的 Euler 方法

在（10.3.7）中，若迭代只进行一次，$y_{n+1}^{[1]}$即作为y_{n+1}之值，这种计算方法称为改进的 Euler 方法，即

$$\begin{cases} \overline{y}_{n+1} = y_n + hf(x_n, y_n) \\ y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, \overline{y}_{n+1})] \end{cases}$$

或写成如下形式：

10.3.8 改进的 Euler 方法/Modified Euler Method

$$y_{n+1} = y_n + \frac{h}{2}[f(x_n, y_n) + f(x_{n+1}, y_n + hf(x_n, y_n))]$$

改进的 Euler 方法是一种形为（10.2.1）的显式单步法，其中

$$\phi(x, y, h) = \frac{1}{2}[f(x, y) + f(x+h, y+hf(x, y))]$$

它的局部截断误差为

$$T_{n+1} = O(h^3)$$

所以，改进的 Euler 方法是一种二阶的方法．

10.3.9 例 用改进的 Euler 方法求解初值问题

$$\frac{dy}{dx}=-y+x+1,\quad 0\leqslant x\leqslant 1$$

$$y(0)=1$$

取步长 $h=0.1$．

解 按公式（10.3.8）计算，对于本例，可化成

$$y_{n+1}=0.905y_n+0.0095n+0.1\quad(n=0,1,\cdots,9)$$

计算结果与准确解 $y(x)=x+e^{-x}$的比较如下表．

x_n	y_n	$y(x_n)$	$\vert y(x_n)-y_n\vert$
0	1.000000	1.000000	0
0.1	1.005000	1.004837	1.63×10^{-4}
0.2	1.019025	1.018731	2.94×10^{-4}
0.3	1.041218	1.040818	4.00×10^{-4}
0.4	1.070802	1.070320	4.82×10^{-4}
0.5	1.107076	1.106531	5.45×10^{-4}
0.6	1.149404	1.148812	5.92×10^{-4}
0.7	1.197211	1.196585	6.26×10^{-4}
0.8	1.249976	1.249329	6.47×10^{-4}
0.9	1.307228	1.306570	6.58×10^{-4}
1.0	1.368541	1.367879	6.62×10^{-4}

10.4

10.4 Runge-Kutta方法

10.4.1 Runge-Kutta（龙格—库塔）方法的一般形式

Runge－Kutta方法是一种 显式 单步方法．R级（R-stage）的 Runge－Kutta 方法形式为

10.4.1 $y_{n+1}=y_n+h\phi(x_n,y_n,h)\quad(n=0,1,\cdots)$

其中

10.4.2 $\phi(x,y,h)=\sum_{r=1}^{R}c_rK_r$

10.4.3 $K_1=f(x,y)$

$$K_r=f(x+ha_r,y+h\sum_{s=1}^{r-1}b_{rs}K_s)\quad(r=2,3,\cdots,R)$$

若希望得到 R 级的 Runge－Kutta 方法是 R 阶的，则要确定系数 c_r，a_r和 b_{rs}，使得局部截断误差为 $O(h^{R+1})$，即假设$y_n=y(x_n)$，将 $y(x_{n+1})$ 及（10.4.1）的 y_{n+1}按 h的幂展开，有 $y(x_{n+1})-y_{n+1}=O(h^{R+1})$。$y(x_{n+1})$ 的展开式为

$$y(x_{n+1})=y(x_n)+hy'(x_n)+\frac{h^2}{2!}y''(x_n)+\cdots+\frac{h^p}{p!}y^{(p)}(x_n)+O(h^{p+1})$$

其中

$$y'(x)=f(x,y)$$

$$y''(x)=\frac{\partial f}{\partial x}+f\frac{\partial f}{\partial y}$$

$$y'''(x)=\frac{\partial^2 f}{\partial x^2}+2f\frac{\partial^2 f}{\partial x\partial y}+f^2\frac{\partial^2 f}{\partial y^2}+\frac{\partial f}{\partial y}\left(\frac{\partial f}{\partial x}+f\frac{\partial f}{\partial y}\right)$$

……

而 y_{n+1}的展开式，按(10.4.1)，其中 $\phi(x,y,h)$按(10.4.2)和（10.4.3）展开，式中的 K_r，K_s按二元函数的 Taylor公式展开。这样可以构造出不同阶的 Runge-Kutta 方法。

10.4.2 二阶 Runge－Kutta 方法

在（10.4.1)—(10.4.3）中，设 $R=2$，按照上述方法展开，有

$$y(x_{n+1})=y(x_n)+hf(x_n,y(x_n))+\frac{h^2}{2}\left[\frac{\partial f(x_n,y(x_n))}{\partial x}\right.$$

$$\left.+f(x_n,y(x_n))\frac{\partial f(x_n,y(x_n))}{\partial y}\right]+O(h^3)$$

$$y_{n+1}=y_n+h(c_1+c_2)f(x_n,y_n)+h^2\left[a_2c_2\frac{\partial f(x_n,y_n)}{\partial x}\right.$$

$$\left.+b_{21}c_2f(x_n,y_n)\frac{\partial f(x_n,y_n)}{\partial y}\right]+O(h^3)$$

设 $y(x_n)=y_n$，欲使 $y(x_{n+1})-y_{n+1}=O(h^3)$，就有

10.4.4
$$\begin{cases} c_1+c_2=1 \\ a_2c_2=\dfrac{1}{2} \\ b_{21}c_2=\dfrac{1}{2} \end{cases}$$

这是四个未知数 c_1，c_2，a_2，b_{21}三个方程的方程组，可以得到多组不同的解。将其解代回（10.4.1)—(10.4.3)，就构成不同的二阶 Runge—Kutta 方法。

10.4.5 例/中点方法/Midpoint Method 在（10.4.4）中，令 $c_1=0$，$c_2=1$，$a_2=b_{21}=\frac{1}{2}$，则（10.4.1）成为

$$y_{n+1}=y_n+hf(x_n+\frac{h}{2},\ y_n+\frac{h}{2}f(x_n,y_n))$$

它相当于在（10.3.2）中的积分式中用矩形数值积分方法近似而得到的公式。

10.4.6 例/改进 Euler 方法 在(10.4.4)中，令$c_1=c_2=\frac{1}{2}$，$a_2=b_{21}=1$，则（10.4.1）成为

$$y_{n+1}=y_n+\frac{h}{2}[f(x_n,y_n)+f(x_n+h,y_n+hf(x_n,y_n))]$$

此即（10.3.8）的改进 Euler 方法。

10.4.7 例/Heun（休恩）方法/Heun Method 在（10.4.4）中，令 $c_1=\frac{1}{4}$，$c_2=\frac{3}{4}$，$a_2=b_{21}=\frac{2}{3}$，则（10.4.1）成为

$$y_{n+1}$$

$$=y_n+\frac{h}{4}\left[f(x_n,y_n)+3f\left(x_n+\frac{2}{3}h,y_n+\frac{2}{3}hf(x_n,y_n)\right)\right]$$

10.4.3 三阶 Runge－Kutta 方法

在（10.4.1）—（10.4.3）中，设 $R=3$，$p=3$，按照上述方法推导，得到

$$\text{10.4.8}\quad\begin{cases}c_1+c_2+c_3=0\\ a_2=b_{21}\\ a_3=b_{31}+b_{32}\\ c_2a_2+c_3a_3=\frac{1}{2}\\ c_2a_2^2+c_3a_3^2=\frac{1}{3}\\ c_3a_2b_{32}=\frac{1}{6}\end{cases}$$

这是八个未知数，六个方程的方程组，可以得到多组不同的解。将满足此方程组的 c_r，a_r,b_{rr}代回(10.4.1)—(10.4.3)，即构成不同的三阶 Runge－Kutta 方法。

10.4.9 例/Heun（休恩）三阶方法/Heun's Third－Order Method 在(10.4.8）中，令 $c_1=\frac{1}{4}$，$c_2=0$，$c_3=\frac{3}{4}$，

$a_2=b_{21}=\frac{1}{3}$，$b_{31}=0$，$a_3=b_{32}=\frac{2}{3}$，则得到

$$y_{n+1}=y_n+\frac{h}{4}(K_1+3K_3)$$

其中

$$K_1=f(x_n,y_n)$$

$$K_2=f(x_n+\frac{h}{3},\ y_n+\frac{h}{3}K_1)$$

$$K_3=f(x_n+\frac{2h}{3},\ y_n+\frac{2h}{3}K_2)$$

10.4.10 例/Kutta三阶方法/Kutta's Third-Order Method 在(10.4.8)中，令 $c_1=\frac{1}{6}$, $c_2=\frac{2}{3}$, $c_3=\frac{1}{6}$, $a_2=b_{21}=\frac{1}{2}$, $a_3=1$, $b_{31}=-1$, $b_{32}=2$，则得到

$$y_{n+1}=y_n+\frac{h}{6}(K_1+4K_2+K_3)$$

其中

$$K_1=f(x_n,y_n)$$

$$K_2=f(x_n+\frac{h}{2},\ y_n+\frac{h}{2}K_1)$$

$$K_3=f(x_n+h,y_n-hK_1+2hK_2)$$

10.4.4 四阶 Runge-Kutta 方法

类似上述方法可以构造多种四级四阶的 Runge-Kutta 方法，其中最经常使用的是经典Runge-Kutta 方法

10.4.11 例/经典 Runge-Kutta 方法/ Classical Runge-Kutta Method

$$y_{n+1}=y_n+\frac{h}{6}(K_1+2K_2+2K_3+K_4)$$

其中

$$K_1=f(x_n,y_n)$$

$$K_2 = f(x_n + \frac{1}{2}h,\ y_n + \frac{1}{2}hK_1)$$

$$K_3 = f(x_n + \frac{1}{2}h,\ y_n + \frac{1}{2}hK_2)$$

$$K_4 = f(x_n + h, y_n + hK_3)$$

10.4.12 **例** 用经典 Runge－Kutta 方法解初值问题

$$\frac{dy}{dx} = -y + x + 1, \quad 0 \leqslant x \leqslant 1$$

$$y(0) = 1$$

取步长 $h = 0.1$。

解 计算结果以及与准确解的比较如下表：

x_n	y_n	$y(x_n)$	$\lvert y(x_n) - y_n \rvert$
0	1.00000000	1.00000000	0
0.1	1.00483750	1.00483742	8×10^{-8}
0.2	1.01873090	1.01873075	1.5×10^{-7}
0.3	1.04081842	1.04081822	2.0×10^{-7}
0.4	1.07032029	1.07032005	2.4×10^{-7}
0.5	1.10653093	1.10653066	2.7×10^{-7}
0.6	1.14881193	1.14881164	2.9×10^{-7}
0.7	1.19658562	1.19658530	3.2×10^{-7}
0.8	1.24932929	1.24932896	3.3×10^{-7}
0.9	1.30656999	1.30656966	3.3×10^{-7}
1.0	1.36787977	1.36787944	3.3×10^{-7}

四阶 Runge-Kutta 方法还有其它方法，如

10.4.13 **例/Gill（基尔）方法/Gill Method**

$$y_{n+1} = y_n + \frac{h}{6}[K_1 + (2-\sqrt{2})K_2 + (2+\sqrt{2})K_3 + K_4]$$

其中

$K_1=f(x_n,y_n)$

$K_2=f(x_n+\frac{h}{2},\ y_n+\frac{h}{2}K_1)$

$K_3=f(x_n+\frac{h}{2},\ y_n+\frac{\sqrt{2}-1}{2}hK_1+(1-\frac{\sqrt{2}}{2})hK_2)$

$K_4=f(x_n+h, y_n-\frac{\sqrt{2}}{2}hK_2+(1+\frac{\sqrt{2}}{2})hK_3)$

10.4.14 例 $y_{n+1}=y_n+\frac{h}{8}(K_1+3K_2+3K_3+K_4)$

其中

$K_1=f(x_n,y_n)$

$K_2=f(x_n+\frac{1}{3}h,\ y_n+\frac{1}{3}hK_1)$

$K_3=f(x_n+\frac{2}{3}h,\ y_n-\frac{1}{3}h(K_1-3K_2))$

$K_4=f(x_n+h, y_n+h(K_1-K_2+K_3))$

10.4.5 高阶 Runge－Kutta 方法

R 级的 Runge－Kutta 方法，在 $R=2$，3，4 时，可以分别得到二、三、四阶的方法，但是,通常 R 级的方法不一定是 R 阶的。设 $p(R)$ 为 R 级方法可以达到的最大阶数，可以证明：

$p(R)=R,\quad R=1,2,3,4$

$p(5)=4,\quad p(6)=5,\quad p(7)=6,\quad p(8)=6,\quad p(9)=7$

$p(R)\leqslant R-2,\quad R=10,11,\cdots$

10.4.15 例/Kutta－Nyström（库塔一尼斯特龙)五阶六级方法

$y_{n+1}=y_n+\frac{h}{192}(23K_1+125K_3-81K_5+125K_6)$

其中

$K_1=f(x_n,y_n)$

$$K_2=f(x_n+\frac{1}{3}h,\ y_n+\frac{1}{3}hK_1)$$

$$K_3=f(x_n+\frac{2}{5}h,\ y_n+\frac{1}{25}h(4K_1+6K_2))$$

$$K_4=f(x_n+h,y_n+\frac{1}{4}h(K_1-12K_2+15K_3))$$

$$K_5=f(x_n+\frac{2}{3}h,\ y_n+\frac{1}{81}h(6K_1+90K_2-50K_3+8K_4))$$

$$K_6=f(x_n+\frac{4}{5}h,\ y_n+\frac{1}{75}h(6K_1+36K_2+10K_3+8K_4))$$

10.4.16 例/Huta（休塔）六阶八级方法

$$y_{n+1}=y_n+\frac{h}{840}(41K_1+216K_3+27K_4+272K_5+27K_6$$
$$+216K_7+41K_8)$$

其中

$$K_1=f(x_n,y_n)$$

$$K_2=f(x_n+\frac{1}{9}h,\ y_n+\frac{1}{9}hK_1)$$

$$K_3=f(x_n+\frac{1}{6}h,\ y_n+\frac{1}{24}h(K_1+3K_2))$$

$$K_4=f(x_n+\frac{1}{3}h,\ y_n+\frac{1}{6}h(K_1-3K_2+4K_3))$$

$$K_5=f(x_n+\frac{1}{2}h,\ y_n+\frac{1}{8}h(-5K_1+27K_2-24K_3+6K_4))$$

$$K_6=f(x_n+\frac{2}{3}h,\ y_n+\frac{1}{9}h(221K_1-981K_2+867K_3$$
$$-102K_4+K_5))$$

$$K_7=f(x_n+\frac{5}{6}h,\ y_n+\frac{1}{48}h(-183K_1+678K_2$$
$$-472K_3-66K_4+80K_5+3K_6))$$

$$K_8=f(x_n+h,y_n+\frac{1}{82}h(716K_1-2079K_2+1002K_3$$
$$+834K_4-454K_5-9K_6+72K_7))$$

10.4.6 Runge－Kutta－Fehlberg（龙格-库塔-费尔贝格）方法

如果要求一个显式单步方法在各节点的整体截断误差 $|y(x_n)-y_n|$ （$n=0, 1, 2, \cdots$）不超过允许的值，而所用节点的数目尽量地小，这就是控制误差的方法。这样所用的节点当然不一定是等步长的。实际上整体截断误差不好确定，但由定理(10.2.7)知，若用一个 p 阶的方法，即局部截断误差为 $T_{n+1}=O(h^{p+1})$，在满足定理的条件下，整体截断误差为 $O(h^p)$，所以考虑用 $\frac{T_{n+1}}{h}$ 代替整体截断误差作估计。

设有一种 p 阶的方法，$T_{n+1}=O(h^{p+1})$，从 x_n 到 x_{n+1} 取步长为 h_n，方法写成

10.4.17 $\quad y_{n+1}=y_n+h_n\phi(x_n, y_n, h_n)$

另有一种高一阶的方法，$\widetilde{T}_{n+1}=O(h^{p+2})$，从 x_n 到 x_{n+1} 取步长 $\widetilde{h}_n$，方法写成

10.4.18 $\quad \widetilde{y}_{n+1}=\widetilde{y}_n+\widetilde{h}_n\widetilde{\phi}(x_n, \widetilde{y}_n, \widetilde{h}_n)$

设计算 y_n 这步没有误差，即 $y(x_n)=y_n=\widetilde{y}_n$，不难看出，若取 $h_n=\widetilde{h}_n$，有

$$T_{n+1}=\widetilde{T}_{n+1}+(\widetilde{y}_{n+1}-y_{n+1})$$

而 $\widetilde{T}_{n+1}$ 是比 T_{n+1} 高一阶的，故有

$$\widetilde{y}_{n+1}-y_{n+1}\approx T_{n+1}\approx Kh^{p+1}$$

得到估计式

$$\frac{T_{n+1}}{h}\approx\frac{\widetilde{y}_{n+1}-y_{n+1}}{h}\approx Kh^p$$

所以在第 n 步的计算中，可以先定一个 h 值，分别用（10.4.17）和（10.4.18）计算出 $\widetilde{y}_{n+1}$ 和 y_{n+1}，然后以 qh 代替 h 计算，这里 q 是一个有界的正数。于是可得估计式

$$\frac{T_{n+1}(qh)}{qh} \approx K(qh)^p \approx q^p \frac{\widetilde{y}_{n+1} - y_{n+1}}{h}$$

因此，只要选择 q 满足

10.4.19 $$q \leqslant \left(\frac{\varepsilon h}{|\widetilde{y}_{n+1} - y_{n+1}|} \right)^{1/p}$$

就可保证

$$\left| \frac{T_{n+1}(qh)}{qh} \right| \leqslant \varepsilon$$

这样，以新的步长 qh 重新计算 y_{n+1}之值，同时，qh 也可以作为下一步步长的初始选择。

10.4.20 Runge－Kutta－Fehlberg 四阶方法 (10.4.17) 和 (10.4.18) 分别取为如下的四阶及五阶公式：

10.4.21 $$y_{n+1} = y_n + h\left(\frac{25}{216}K_1 + \frac{1408}{2565}K_3 + \frac{2197}{4104}K_4 - \frac{1}{5}K_5 \right)$$

10.4.22 $$\widetilde{y}_{n+1} = y_n + h\left(\frac{16}{135}K_1 + \frac{6656}{12825}K_3 + \frac{28561}{56430}K_4 - \frac{9}{50}K_5 + \frac{2}{55}K_6 \right)$$

其中

$$K_1 = f(x_n, y_n)$$

$$K_2 = f\left(x_n + \frac{h}{4},\ y_n + \frac{h}{4}K_1 \right)$$

$$K_3 = f\left(x_n + \frac{3h}{8},\ y_n + \frac{3h}{32}K_1 + \frac{9h}{32}K_2 \right)$$

$$K_4 = f\left(x_n + \frac{12h}{13},\ y_n + \frac{1932h}{2197}K_1 - \frac{7200h}{2197}K_2 + \frac{7296h}{2197}K_3 \right)$$

$$K_5 = f\left(x_n + h, y_n + \frac{439h}{216}K_1 - 8hK_2 + \frac{3680h}{513}K_3 - \frac{845h}{4104}K_4 \right)$$

$$K_6=f\left(x_n+\frac{h}{2},\ y_n-\frac{8h}{27}K_1+2hK_2-\frac{3544h}{2565}K_3\right.$$
$$\left.+\frac{1859h}{4104}K_4-\frac{11h}{40}K_5\right)$$

这里（10.4.21）和（10.4.22）都是Runge－Kutta公式，尽管分别是四阶和五阶的，但所用到K_i（$i=1$，…，5)都是相同的，满足（10.4.19）的q一般取为

10.4.23 $$q=\left(\frac{\varepsilon h}{2\left|\widetilde{y}_{n+1}-y_{n+1}\right|}\right)^{1/4}=0.84\left(\frac{\varepsilon h}{\left|\widetilde{y}_{n+1}-y_{n+1}\right|}\right)^{1/4}$$

在具体计算时，先选定一个h（或取为上一步的h)，计算

$$\frac{\left|\widetilde{y}_{n+1}-y_n\right|}{h}$$
$$=\left|\frac{1}{360}K_1-\frac{128}{4275}K_3-\frac{2197}{75240}K_4+\frac{1}{50}K_5+\frac{2}{55}K_6\right|$$

若$\dfrac{\left|\widetilde{y}_{n+1}-y_n\right|}{h}<\varepsilon$，则按（10.4.21)计算$y_{n+1}$，转入下一步.

若$\dfrac{\left|\widetilde{y}_{n+1}-y_n\right|}{h}\geqslant\varepsilon$，则按（10.4.23）计算$q$，再以$qh$作为新的步长重新计算.

10.4.24 Runge－Kutta－Fehlberg五阶方法 （10.4.17）和（10.4.18）分别取为

$$y_{n+1}=y_n+h\sum_{k=1}^{6}c_kf_k$$

$$\widetilde{y}_{n+1}=y_n+h\sum_{k=1}^{8}\widetilde{c}_kf_k$$

其中

$$f_k=f\left(x_n+\alpha_kh,\ y_n+h\sum_{\lambda=1}^{k-1}\beta_{k\lambda}f_\lambda\right)$$

式中的系数如下表，而和式$\sum\limits_{\lambda=1}^{0}$认为是0。

k \ λ	$\beta_{k\lambda}$ 1	2	3	4	5	6	7	α_k	c_k	$\tilde{c}_k$
1	0							0	$\frac{31}{380}$	$\frac{7}{1408}$
2	$\frac{1}{6}$							$\frac{1}{6}$	0	0
3	$\frac{4}{75}$	$\frac{16}{75}$						$\frac{4}{15}$	$\frac{1125}{2816}$	$\frac{1125}{2816}$
4	$\frac{5}{6}$	$-\frac{8}{3}$	$\frac{5}{2}$					$\frac{2}{3}$	$\frac{9}{32}$	$\frac{9}{32}$
5	$-\frac{8}{5}$	$\frac{144}{25}$	-4	$\frac{16}{25}$				$\frac{4}{5}$	$\frac{125}{768}$	$\frac{125}{768}$
6	$\frac{361}{320}$	$-\frac{18}{5}$	$\frac{407}{128}$	$-\frac{11}{80}$	$\frac{55}{128}$			1	$\frac{5}{66}$	0
7	$-\frac{11}{640}$	0	$\frac{11}{256}$	$-\frac{11}{160}$	$\frac{11}{256}$	0		0	—	$\frac{5}{66}$
8	$\frac{93}{640}$	$-\frac{18}{5}$	$\frac{803}{256}$	$-\frac{11}{160}$	$\frac{99}{256}$	0	1	1	—	$\frac{5}{66}$

10.4.25 Runge-Kutta-Fehlberg 六阶方法 (10.4.17) 和 (10.4.18) 分别取为

$$y_{n+1}=y_n+h\sum_{k=1}^{8}c_k f_k$$

$$\tilde{y}_{n+1}=y_n+h\sum_{k=1}^{10}\tilde{c}_k f_k$$

其中 f_k的表达式同五阶方法，系数如表(见588页)。

10.4.26 Runge - Kutta - Fehlberg 七阶方法　(10.4.17) 和 (10.4.18) 分别取为

$$y_{n+1}=y_n+\sum_{k=1}^{11}c_kf_k$$

$$\widetilde{y}_{n+1}=y_n+\sum_{k=1}^{13}\widetilde{c_k}f_k$$

其中 f_k表达式同五阶方法，系数如下表.

k \ λ	$\beta_{k\lambda}$						
	1	2	3	4	5	6	7
1	0						
2	$\frac{2}{27}$						
3	$\frac{1}{36}$	$\frac{1}{12}$					
4	$\frac{1}{24}$	0	$\frac{1}{8}$				
5	$\frac{5}{12}$	0	$-\frac{25}{16}$				
6	$\frac{1}{20}$	0	0	$\frac{1}{4}$	$\frac{1}{5}$		
7	$-\frac{25}{108}$	0	0	$\frac{125}{108}$	$-\frac{65}{27}$	$\frac{125}{54}$	
8	$\frac{31}{300}$	0	0	0	$\frac{61}{225}$	$-\frac{2}{9}$	$\frac{13}{900}$
9	2	0	0	$-\frac{53}{6}$	$\frac{704}{45}$	$-\frac{107}{9}$	$\frac{67}{90}$
10	$-\frac{91}{108}$	0	0	$\frac{23}{108}$	$-\frac{976}{135}$	$\frac{311}{54}$	$-\frac{19}{60}$
11	$\frac{2383}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{301}{82}$	$\frac{2133}{4100}$
12	$\frac{3}{205}$	0	0	0	0	$-\frac{6}{41}$	$-\frac{3}{205}$
13	$-\frac{1777}{4100}$	0	0	$-\frac{341}{164}$	$\frac{4496}{1025}$	$-\frac{289}{82}$	$\frac{2193}{4100}$

8	9	10	11	12	α_k	c_k	$\tilde{c}_k$
					0	$\frac{41}{840}$	0
					$\frac{2}{27}$	0	0
					$\frac{1}{9}$	0	0
					$\frac{1}{6}$	0	0
					$\frac{5}{12}$	0	0
					$\frac{1}{2}$	$\frac{34}{105}$	$\frac{34}{105}$
					$\frac{5}{6}$	$\frac{9}{35}$	$\frac{9}{35}$
					$\frac{1}{6}$	$\frac{9}{35}$	$\frac{9}{35}$
3					$\frac{2}{3}$	$\frac{9}{280}$	$\frac{9}{280}$
$\frac{17}{6}$	$-\frac{1}{12}$				$\frac{1}{3}$	$\frac{9}{280}$	$\frac{9}{280}$
$\frac{45}{82}$	$\frac{45}{164}$	$\frac{18}{41}$			1	$\frac{41}{840}$	0
$-\frac{3}{41}$	$\frac{3}{41}$	$\frac{6}{41}$	0		0	—	$\frac{41}{840}$
$\frac{51}{82}$	$\frac{33}{164}$	$\frac{12}{41}$	0	1	1	—	$\frac{41}{840}$

k \ λ	$\beta_{k\lambda}$ 1	2	3	4	5	6	7	8	9	α_k	c_k	$\tilde{c}_k$
1	0									0	$\frac{77}{1440}$	$\frac{11}{864}$
2	$\frac{2}{33}$									$\frac{2}{33}$	0	0
3	0	$\frac{4}{33}$								$\frac{4}{33}$	0	0
4	$\frac{1}{22}$	0	$\frac{3}{22}$							$\frac{2}{11}$	$\frac{1771561}{6289920}$	$\frac{1771561}{6289920}$
5	$\frac{43}{64}$	0	$-\frac{16[illegible]}{64}$	$\frac{77}{32}$						$\frac{1}{2}$	$\frac{32}{105}$	$\frac{32}{105}$
6	$-\frac{2383}{486}$	0	$\frac{1067}{54}$	$-\frac{26312}{1701}$	$\frac{2176}{1701}$					$\frac{2}{3}$	$\frac{243}{2560}$	$\frac{243}{2560}$
7	$\frac{10077}{4802}$	0	$-\frac{5643}{686}$	$\frac{116259}{16807}$	$-\frac{6240}{16807}$	$\frac{1053}{2401}$				$\frac{6}{7}$	$\frac{16807}{74880}$	$\frac{16807}{74880}$
8	$-\frac{733}{176}$	0	$\frac{141}{8}$	$-\frac{335763}{23296}$	$\frac{216}{77}$	$-\frac{4617}{2816}$	$\frac{7203}{9152}$			1	$\frac{11}{270}$	0
9	$\frac{15}{352}$	0	0	$-\frac{5445}{46592}$	$\frac{18}{77}$	$-\frac{1215}{5632}$	$\frac{1029}{18304}$	0		0	—	$\frac{11}{270}$
10	$-\frac{1833}{352}$	0	$\frac{141}{8}$	$-\frac{51237}{3584}$	$\frac{18}{7}$	$-\frac{729}{512}$	$\frac{1029}{1408}$	0	1	1	—	$\frac{11}{270}$

10.5 线性多步法

10.5.1 线性多步法的一般形式

如（10.1.13）或（10.1.14）所示，解初值问题（10.1.1），（10.1.2）的 k 步线性多步法的一般形式为

10.5.1 $$y_{n+k}=-\alpha_0 y_n-\alpha_1 y_{n+1}-\cdots-\alpha_{k-1}y_{n+k-1}+h(\beta_0 f_n+\beta_1 f_{n+1}+\cdots+\beta_k f_{n+k})$$

可以由 Taylor 展开式或数值积分公式推导.

对于一种线性多步法（10.5.1），可以对应地定义算子

10.5.2 $$\mathscr{L}[y(x);h]=\sum_{i=0}^{k}[\alpha_i y(x+ih)-h\beta_i y'(x+ih)]$$

其中$\alpha_k=1$，$y(x)$ 是$[x_0，b]$上任意的连续可微函数，如果将 $y(x+ih)$ 及$y'(x+ih)$ 按Taylor公式展开：

$$y(x+ih)=y(x)+\frac{ih}{1!}y'(x)+\frac{(ih)^2}{2!}y''(x)+\cdots$$

$$y'(x+ih)=y'(x)+\frac{ih}{1!}y''(x)+\frac{(ih)^2}{2!}y'''(x)+\cdots$$

再代入（10.5.2），可以整理成

10.5.3 $$\mathscr{L}[y(x);h]=c_0 y(x)+c_1 hy'(x)+\cdots+c_q h^q y^{(q)}(x)+\cdots$$

其中

10.5.4 $$\begin{cases} c_0=\alpha_0+\alpha_1+\alpha_2+\cdots+\alpha_k \\ c_1=\alpha_1+2\alpha_2+\cdots+k\alpha_k-(\beta_0+\beta_1+\beta_2+\cdots+\beta_k) \\ c_q=\dfrac{1}{q!}(\alpha_1+2^q\alpha_2+\cdots+k^q\alpha_k) \\ \qquad -\dfrac{1}{(q-1)!}(\beta_1+2^{q-1}\beta_2+\cdots+k^{q-1}\beta_k) \\ \qquad (q=2,3,\cdots) \end{cases}$$

10.5.5 定义 /阶/Order 若在(10.5.3)式中，$c_0=c_1=\cdots=c_p=0$，$c_{p+1}\neq 0$，则称算子（10.5.2）及对应的线性多步法（10.5.1）是 p 阶的。

10.5.6 定义/相容性/Consistency 若线性多步法(10.5.1)的阶 $p\geqslant 1$，则称它与初值问题（10.1.1），（10.1.2）是相容的。

10.5.7 定理 k 步线性多步法(10.5.1)与初值问题(10.1.1)，(10.1.2）相容的充分必要条件是：

$$\sum_{i=0}^{k}\alpha_i=0,\quad \sum_{i=1}^{k} i\alpha_i=\sum_{i=0}^{k}\beta_i$$

实际使用的方法应该是相容的，所以要寻找系数 α_0，$\cdots\alpha_k$，β_0，$\cdots\beta_k$，使得 $c_0=c_1=\cdots=c_p=0$，$c_{p+1}\neq 0$（其中 $p\geqslant 1$）得到的系数代入（10.5.1），即可构成一个 p 阶的线性多步法（$k=1$ 时为单步法）。当 $\beta_k=0$ 时，方法是显式的，$\beta_k\neq 0$ 时，方法是隐式的。

10.5.8 定义 /局部截断误差/Local Truncation Error 设 y 是初值问题(10.1.1)，(10.1.2)的准确解，则称 $\mathscr{L}[y(x_n);h]$ 为线性多步法（10.5.1）在 x_{n+k} 的**局部截断误差**，或记为 $T_{n+k}=\mathscr{L}[y(x_n);h]$

根据定义，若线性多步法（10.5.1）是 p 阶的方法，则其局部截断误差为

10.5.9 $T_{n+k}=c_{p+1}h^{p+1}y^{(p+1)}(x_n)+c_{p+2}h^{p+2}y^{(p+2)}(x_n)+\cdots$

10.5.10 定理 如果假设计算 $y(x_{n+k})$ 前 k 步所得离散解是准确的，即设 $y_{n+i}=y(x_{n+i})$，$i=0$，$1\cdots$，$k-1$，则对于显式的线性多步法（$\beta_k=0$），有

$T_{n+k}=y(x_{n+k})-y_{n+k}$

而对于 p 阶的隐式线性多步法（$\beta_k\neq 0$），有

$y(x_{n+k})-y_{n+k}=c_{p+1}h^{p+1}y^{(p+1)}(x_n)+O(h^{p+2})$

亦即，对于隐式的线性多步法，设前 k 步离散解准确时，截断误差（10.5.9）式与 $y(x_{n+k})-y_{n+k}$ 展开式的首项是相同的。

10.5.11 定义/主局部截断误差/Principal Local Truncation

p阶线性多步法局部截断误差T_{n+k}的首项$c_{p+1}h^{p+1}y^{(p+1)}(x_n)$称为**主局部截断误差**。

10.5.12 定义/整体截断误差/Global Truncation Error

$y(x_{n+k})-y_{n+k}$称（10.5.1）在 x_{n+k} 的**整体截断误差**。

10.5.13 例/Euler方法 在(10.5.1）中，令$k=1$，系数取为$\alpha_0=-1$，$\alpha_1=1$，$\beta_0=1$，$\beta_1=0$，则有

$$y_{n+1}=y_n+hf(x_n,y_n)$$

这即是Euler方法(10.3.1).显然，由(10.5.4)．$c_0=c_1=0$，$c_2=\frac{1}{2}$，所以Euler方法是一阶的单步显式方法，其主局部截断误差为$\frac{1}{2}h^2y''(x_n)$。这里关于Euler方法的阶和局部截断误差的分析，与（10.3）的分析是一致的。

类似于（10.1.4），在区间$[x_n, x_{n+k}]$上，初值问题（10.1.1），（10.1.2）的解 $y(x)$满足

10.5.14 $$y(x_{n+k})=y(x_n)+\int_{x_n}^{x_{n+k}}f(t,y(t))dt$$

如果对右端的积分式运用各种数值积分公式计算，可构造出多种计算 $y(x_{n+k})$ 的近似值 y_{n+k} 的计算方法。这些方法称为由数值积分公式推导的方法。

10.5.15 例 /Euler方法 令$k=1$，(10.5.14）化为

$$y(x_{n+1})=y(x_n)+\int_{x_n}^{x_{n+1}}f(t,y(t))dt$$

右端的积分用左矩形公式计算，并设 $y(x_n)=y_n$，就得到

Euler 方法

$$y_{n+1}=y_n+hf(x_n,y_n)$$

10.5.2 Adams（亚当斯）方法

把（10.5.14）写成

10.5.16 $$y(x_{n+1})=y(x_n)+\int_{x_n}^{x_{n+1}}f(t,y(t))dt$$

积分式中$f(t,y(t))$，用k个节点x_n，x_{n-1}，$\cdots$，x_{n-k+1}上的f值f_n，f_{n-1}，$\cdots$，f_{n-k+1}作一个$k-1$次的多项式插值公式

$$f(t,y(t))\approx f_n l_n(t)+f_{n-1}l_{n-1}(t)+\cdots+f_{n-k+1}l_{n-k+1}(t)$$

其中$l_{n-j}(t)$是对应节点上的插值基函数。

这样得到插值公式用到区间$[x_n,\ x_{n+1}]$上，实际上是一种外插公式，把$f(t,\ y(t))$上述的插值逼近代入（10.5.16），得到初值问题（10.1.1），（10.1.2）如下的数值方法：

10.5.17 Adams－Bashforth（亚当斯-巴诗福斯）方法

$$y_{n+1}=y_n+h(\gamma_0 f_n+\gamma_1 f_{n-1}+\cdots+\gamma_{k-1}f_{n-k+1})$$

其中

$$\gamma_j=\frac{1}{h}\int_{x_n}^{x_{n+1}}l_{n-j}(t)dt\qquad(j=0,1,\cdots,k-1)$$

（10.5.17）或称 Adams 显式方法，它是k步的显式线性多步方法，当$k=1$时，（10.5.17）就成为 Euler方法。给定了k，就可计算γ_j，而系数γ_j确定后，（10.5.17)也可写回（10.5.1）的形式，其局部截断误差也可用（10.5.9）计算出来，几个k的系数及主截断误差列在表（10.5.18）中。

10.5.18 Adams-Bashforth方法系数及主局部截断误差表

k	γ_0	γ_1	γ_2	γ_3	γ_4	主局部截断误差
1	1					$\frac{1}{2}h^2y''(x_n)$
2	$\frac{3}{2}$	$-\frac{1}{2}$				$\frac{5}{12}h^3y^{(3)}(x_n)$
3	$\frac{23}{12}$	$-\frac{16}{12}$	$\frac{5}{12}$			$\frac{3}{8}h^4y^{(4)}(x_n)$
4	$\frac{55}{24}$	$-\frac{59}{24}$	$\frac{37}{24}$	$-\frac{9}{24}$		$\frac{251}{720}h^5y^{(5)}(x_n)$
5	$\frac{1901}{720}$	$-\frac{2774}{720}$	$\frac{2616}{720}$	$-\frac{1274}{720}$	$\frac{251}{720}$	$\frac{95}{288}h^6y^{(6)}(x_n)$

(10.5.16) 的积分式中的 $f(t,y(t))$，若用 k 个节点 x_{n+1}，x_n，…，x_{n-k+2} 上的 f 值 f_{n+1}，f_n，…，f_{n-k+2} 为一个 $k-1$ 次的插值多项式作为 $f(t,y(t))$ 的近似，代入 (10.5.16) 后，就得到解初值问题 (10.1.1)，(10.1.2) 如下的数值方法：

10.5.19 Adams－Moulton（亚当斯-莫尔顿）方法

$$y_{n+1}=y_n+h(\gamma_0 f_{n+1}+\gamma_1 f_n+\cdots+\gamma_{k-1}f_{n-k+2})$$

其中

$$\gamma_j=\frac{1}{h}\int_{x_n}^{x_{n+1}} l_{n+1-j}(t)dt \qquad (j=0,1,\cdots,k-1)$$

$l_{n+1-j}(t)$ 为节点 x_{n+1-j} 上的插值基函数.

(10.5.19) 也可以称为Adams隐式方法. 当 $k=1$ 时，即为隐式的 Euler 方法；当 $k=2$ 时，即为梯形方法，当 $k\geqslant 2$ 时，是 $k-1$ 步的隐式方法，可以计算出系数 γ_j 及对应的主局部截断误差. 如表 (10.5.20) 所示.

10.5.20 Adams－Moulton 方法系数及主局部截断误差表

k	γ_0	γ_1	γ_2	γ_3	γ_4	主局部截断误差
1	1					$-\frac{1}{2}h^2y''(x_n)$
2	$\frac{1}{2}$	$\frac{1}{2}$				$-\frac{1}{12}h^3y^{(3)}(x_n)$
3	$\frac{5}{12}$	$\frac{8}{12}$	$-\frac{1}{12}$			$-\frac{1}{24}h^4y^{(4)}(x_n)$
4	$\frac{9}{24}$	$\frac{19}{24}$	$-\frac{5}{24}$	$\frac{1}{24}$		$-\frac{19}{720}h^5y^{(5)}(x_n)$
5	$\frac{251}{720}$	$\frac{646}{720}$	$-\frac{264}{720}$	$\frac{106}{720}$	$-\frac{19}{720}$	$-\frac{3}{160}h^6y^{(6)}(x_n)$

$k=4$的Adams－Bashforth方法和 Adams－Moulton 方法，是经常使用的四阶的方法，但前者是四步而后者是三步的。它们可分别写成

$$y_{n+1}=y_n+\frac{h}{24}[55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}]$$

$$y_{n+1}=y_n+\frac{h}{24}[9f_{n+1}+19f_n-5f_{n-1}+f_{n-2}]$$

10.5.21 例 用四阶的Adams-Bashforth方法及四阶的 Adams-Moulton 方法解初值问题

$$\frac{dy}{dx}=-y+x+1 \quad , \quad 0\leqslant x\leqslant 1$$

$y(0)=1$

步长取为 $h=0.1$。

解 所用到的初始值，分别用准确解 $y(x)=e^{-x}+x$ 在节点之值计算，计算结果如 595 页表。

从计算结果可见，四阶的Adams－Moulton方法的误差要比四阶的Adams－Bashforth方法的误差小。

x_n	Adams—Bashforth		Adams—Moulton	
	y_n	误 差	y_n	误 差
0.3			1.04081801	2.1×10^{-7}
0.4	1.07032292	2.87×10^{-6}	1.07031966	3.9×10^{-7}
0.5	1.10653548	4.82×10^{-6}	1.10653014	5.2×10^{-7}
0.6	1.14881841	6.77×10^{-6}	1.14881101	6.3×10^{-7}
0.7	1.19659339	8.09×10^{-6}	1.19658459	7.1×10^{-7}
0.8	1.24933816	9.19×10^{-6}	1.24932819	7.7×10^{-7}
0.9	1.30657961	9.95×10^{-6}	1.30656885	8.1×10^{-7}
1.0	1.36788996	1.05×10^{-5}	1.36787860	8.4×10^{-7}

10.5.3 Milne（米尔尼）方法

Milne 方法是一种显式四阶多步法，可以看成在积分恒等式

$$y(x_{n+1})=y(x_{n-3})+\int_{x_{n-3}}^{x_{n+1}} f(t,y(t))dt$$

中，用x_{n-2}，x_{n-1}，x_n三个节点作插值公式代替$f(t,y(t))$后而得，其为：

10.5.22 Milne 方法

$$y_{n+1}=y_{n-3}+\frac{4h}{3}(2f_n-f_{n-1}+2f_{n-2})$$

Milne方法的主局部截断误差是 $\frac{14}{45}h^5y^{(5)}(x_n)$。类似地，对于积分恒等式

$$y(x_{n+1})=y(x_{n-1})+\int_{x_{n-1}}^{x_{n+1}} f(t,y(t))dt$$

用x_{n-1}，x_n，x_{n+1}三个节点作Simpson数值积分公式，得到：

10.5.23 Simpson（辛甫生）方法

$$y_{n+1}=y_{n-1}+\frac{h}{3}(f_{n+1}+4f_n+f_{n-1})$$

(10.5.23) 有时也称为Milne方法，它的主局部截断误差是 $-\frac{1}{90}h^5y^{(5)}(x_n)$.

10.5.4 Hamming（哈明）方法

Hamming方法是一种四阶隐式方法，它可用 Taylor 展开式的方法推导，其为

10.5.24 Hamming 方法

$$y_{n+1}=\frac{1}{8}(9y_n-y_{n-2})+\frac{3h}{8}(f_{n+1}+2f_n-f_{n-1})$$

Hamming方法的主局部截断误差为 $-\frac{h^5}{40}y^{(5)}(x_n)$.

10.6

10.6 预测—校正方法

10.6.1 预测—校正的一般方法

一个隐式的线性多步法，可由（10.5.1）写成

10.6.1 $$y_{n+k}+\sum_{j=0}^{k-1}\alpha_jy_{n+j}=h\beta_kf(x_{n+k},y_{n+k})+h\sum_{j=0}^{k-1}\beta_jf_{n+j}$$

其中 $\beta_k\neq0$. 它可以根据（10.1.21）的迭代法来求解，即

10.6.2 $$y_{n+k}^{[s+1]}+\sum_{j=0}^{k-1}\alpha_jy_{n+j}=h\beta_kf(x_{n+k},y_{n+k}^{[s]})+h\sum_{j=0}^{k-1}\beta_jf_{n+j}$$

$$(s=0,1,2,\cdots)$$

在满足定理（10.1.22）的条件下，迭代对任意的初值 $y_{n+k}^{[0]}$ 收敛。

（10.6.2）的每一步，均要计算一次函数值 $f(x_{n+k},y_{n+k}^{[s]})$ ，所以迭代计算要多次调用计算函数值 $f(x,y)$ 的

子程序，这样要花费较大的代价，因此可以考虑用另外一个显式的方法作出 y_{n+k} 的估计（称为预测），代入隐式方法（10.6.1）的右边，得到 y_{n+k} 之值（称为校正），这就组成预测—校正方法。虽然（10.6.2）的迭代法也可以看成是不断进行的校正过程，但它的迭代次数是由 $|y_{n+k}^{[s+1]}-y_{n+k}^{[s]}|<\varepsilon$ 来确定，而预测—校正方法则是事先规定好了预测与校正的次数。

10.6.3 例/改进的 Euler 方法 （10.3.8）的改进 Euler 方法，可以看成由以下两步组成：

预测：$y_{n+1}^{[0]}=y_n+hf(x_n,y_n)$

校正：$y_{n+1}=y_n+\dfrac{h}{2}[f(x_n,y_n)+f(x_{n+1},y_{n+1}^{[0]})]$

其中预测值是由 Euler 公式得到，校正值是由隐式的梯形公式作一次迭代得到。

在预测—校正方法中，除预测、校正两步外，有时也可加上计算函数值的步骤。例如改进Euler方法可以写成：

10.6.4 改进的Euler方法

预测：$y_{n+1}^{[0]}=y_n+hf(x_n,y_n)$

计算：$m_{n+1}=f(x_{n+1},y_{n+1}^{[0]})$

校正：$y_{n+1}=y_n+\dfrac{h}{2}[f(x_n,y_n)+m_{n+1}]$

这里 m_{n+1} 是直接计算 $f(x_{n+1},y_{n+1}^{[0]})$ 的函数值。有时，这种计算要经过一定的“修正”步骤，使计算精确度得到提高。

10.6.2 Adams 预测—校正方法

分别以四阶的 Adams-Bashforth方法和Adams-Moulton 方法作为预测式和校正式，可得到如下的四阶预测—校

正方法。

10.6.5 Adams 四阶预测—校正方法/Adams Fourth-Order Predictor-Corrector Method

预测：$y_{n+1}^{[0]}=y_n+\frac{h}{24}[55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}]$

计算：$m_{n+1}=f(x_{n+1},y_{n+1}^{[0]})$

校正：$y_{n+1}=y_n+\frac{h}{24}[9m_{n+1}+19f_n-5f_{n-1}+f_{n-2}]$

10.6.6 例 用Adams四阶预测—校正方法解初值问题

$$\frac{dy}{dx}=-y+x+1,\quad 0\leqslant x\leqslant 1$$

$$y(0)=1$$

解 取$h=0.1$，开始值y_1，y_2，y_3用Runge-Kutta方法计算（例10.4.12），结果如下：

x_n	y_n	$\lvert y(x_n)-y_n\rvert$
0.4	1.07031992	1.3×10^{-7}
0.5	1.10653027	3.9×10^{-7}
0.6	1.14881103	6.1×10^{-7}
0.7	1.19658453	7.7×10^{-7}
0.8	1.24932806	9.0×10^{-7}
0.9	1.30656866	1.0×10^{-6}
1.0	1.36787837	1.1×10^{-6}

从（10.5.18）及（10.5.20）可知，四阶Adams-Bashforth方法和Adams-Moulton方法的主局部截断误差分别为$\frac{251}{720}h^5y^{(5)}(x_n)$，$-\frac{19}{720}h^5y^{(5)}(x_n)$，所以对(10.6.5)的预测值和校正值$y_{n+1}^{[0]}$及$y_{n+1}$有：

$$y(x_{n+1})-y_{n+1}^{[0]}\approx\frac{251}{720}h^5y^{(5)}(x_n)$$

$$y(x_{n+1})-y_{n+1}\approx-\frac{19}{720}h^5y^{(5)}(x_n)$$

经过简单的运算，可得事后的估计式：

$$y(x_{n+1})-y_{n+1}^{[0]}\approx-\frac{251}{270}(y_{n+1}^{[0]}-y_{n+1})$$

$$y(x_{n+1})-y_{n+1}\approx\frac{19}{270}(y_{n+1}^{[0]}-y_{n+1})$$

由这两个式子，可对（10.6.5）进一步改进，预测值进一步取为$y_{n+1}^{[0]}-\frac{251}{270}(y_{n+1}^{[0]}-y_{n+1})$，其中$y_{n+1}$在未算出时，$y_{n+1}^{[0]}-y_{n+1}$用$y_n^{[0]}-y_n$代替，而校正值进一步取为$y_{n+1}+\frac{19}{270}(y_{n+1}^{[0]}-y_{n+1})$，于是得到

10.6.7 改进的Adams四阶预测—校正方法

预测：$p_{n+1}=y_n+\frac{h}{24}[55f_n-59f_{n-1}+37f_{n-2}-9f_{n-3}]$

改进：$m_{n+1}=p_{n+1}+\frac{251}{270}(c_n-p_n)$

计算：$m'_{n+1}=f(x_{n+1},m_{n+1})$

校正：$c_{n+1}=y_n+\frac{h}{24}[9m'_{n+1}+19f_n-5f_{n-1}+f_{n-2}]$

改进：$y_{n+1}=c_{n+1}+\frac{19}{270}(p_{n+1}-c_{n+1})$

计算：$f_{n+1}=f(x_{n+1},y_{n+1})$

（10.6.7）的初始值f_i（$i=1,2,3$）同样要另外计

算。开始时的$c_n - p_n$，可令其为0。

10.6.3 Hamming 预测—校正方法

把 Milne公式（10.5.22）与Hamming公式（10.5.24）相配合，并利用其主局部截断误差作出改进，可得到：

10.6.8 **Hamming预测—校正方法**

预测：$p_{n+1} = y_{n-3} + \frac{4h}{3}(2f_n - f_{n-1} + 2f_{n-2})$

改进：$m_{n+1} = p_{n+1} - \frac{112}{121}(p_n - c_n)$

计算：$m'_{n+1} = f(x_{n+1}, m_{n+1})$

校正：$c_{n+1} = \frac{9}{8}y_n - \frac{1}{8}y_{n-2} + \frac{3h}{8}(m'_{n+1} + 2f_n - f_{n-1})$

改进：$y_{n+1} = c_{n+1} + \frac{9}{121}(p_{n+1} - c_{n+1})$

计算：$f_{n+1} = f(x_{n+1}, y_{n+1})$

10.7

10.7 外推方法

10.7.1 外推的一般方法

设$y(x)$为初值问题（10.1.1），（10.1.2）的准确解，取步长为h_i，用某种数值方法计算N_i步，至$x=H$。这里$H = N_i h_i$，记计算的近似值为$y(x_0 + H; h_i)$。如果这种数值方法的整体截断误差为$-(A_1 h_i^\gamma + A_2 h_i^{2\gamma} + A_3 h_i^{3\gamma} + \cdots)$，其中$\gamma$是一个正整数，即假设

10.7.1 $y(x_0 + H; h_i) = y(x_0 + H) + A_1 h_i^\gamma + A_2 h_i^{2\gamma} + A_3 h_i^{3\gamma} + \cdots$

如果分别用二个步长h_0，h_1来计算（其中$h_0 > h_1$），在都计算到$x=H$后，可分别列出（10.7.1)式，消去A_1项，有

10.7.2 $y(x_0+H;h_0)+\dfrac{y(x_0+H;h_0)-y(x_0+H;h_1)}{(h_1/h_0)^\gamma-1}$

$=y(x_0+H)-A_2h_0^\gamma h_1^\gamma+\cdots$

(10.7.2) 的左端就是$y(x_0+H)$ 较好的逼近.

如果用三个步长$h_0>h_1>h_2$计算，h_1和 h_2 的计算得到类似 (10.7.2) 的式子，消去 A_2项，便可得到更高阶的逼近。这个过程还可对$h_0>h_1>h_2>h_3>\cdots$继续下去，得到

10.7.3

$$
\begin{array}{lllll}
y(x_0+H;h_0)=p_0^{(0)} & & & & \\
y(x_0+H;h_1)=p_1^{(0)} & p_0^{(1)} & & & \\
y(x_0+H;h_2)=p_2^{(0)} & p_1^{(1)} & p_0^{(2)} & & \\
y(x_0+H;h_3)=p_3^{(0)} & p_2^{(1)} & p_1^{(2)} & p_0^{(3)} & \\
\vdots & \vdots & \vdots & \vdots & \vdots
\end{array}
$$

以上的值可逐行由下式决定：

10.7.4 $p_i^{(j)}=p_{i+1}^{(j-1)}+\dfrac{p_{i+1}^{(j-1)}-p_i^{(j-1)}}{(h_i/h_{i+j})^\gamma-1}\quad(j=1,2,\cdots)$

可以证明：

10.7.5 $p_i^{(j)}=y(x_0+H)+O(h_i^\gamma h_{i+1}^\gamma\cdots h_{i+j}^\gamma)$

所以每列比其左边的列能更快地收敛到$y(x_0+H)$，每行比其上面的行收敛得快，而对角线比行和列收敛得更快.

使用外推方法的关键是运用 (10.7.1) 的表达式，以Euler法为例，可以证明，若y_n是Euler方法计算的近似值，则有

$y_n=y(x_n)+Ah+O(h^2)$

10.7.6 例 用外推的Euler方法解

$\dfrac{dy}{dx}=-y+x+1$， $0\leqslant x\leqslant1$

$y(0)=1$

解 取$h_0=0.1$，则近似值$y(x_n,0.1)$在例 (10.3.4)

中给出了结果。再取$h_1=0.05$进行计算，在$x=0.1,0.2,\cdots,1$上，得到两种方法的结果，用表（10.7.3）外推一次，由（10.7.4）得

$$p_0^{(1)}=2y(x_0+H;0.05)-y(x_0+H;0.1)$$

计算结果及与（10.3.4）比较误差的大小列表如下。

x	$y(x;0.1)$	$y(x;0.05)$	$p_0^{(1)}$	误差
0	1.000000	1.000000	1.000000	0
0.1	1.000000	1.002500	1.005000	1.63×10^{-4}
0.2	1.010000	1.014506	1.019012	2.81×10^{-4}
0.3	1.029000	1.035092	1.041184	3.66×10^{-4}
0.4	1.056100	1.063420	1.070740	4.20×10^{-4}
0.5	1.090490	1.098737	1.106984	4.53×10^{-4}
0.6	1.131441	1.140360	1.149279	4.67×10^{-4}
0.7	1.178297	1.187675	1.197053	4.68×10^{-4}
0.8	1.230467	1.240126	1.249785	4.56×10^{-4}
0.9	1.287420	1.297214	1.307008	4.38×10^{-4}
1.0	1.348678	1.358485	1.368292	4.13×10^{-4}

10.7.2 Gragg（格拉格）外推方法

使用外推方法时，所用的数值方法必须具有（10.7.1）的性质，且其中γ应尽量大些。利用显式的中点公式

10.7.7 $\quad y_{n+2}=y_n+2hf_{n+1}$

可以得到$\gamma=2$的（10.7.1）展开式。Gragg证明若开始值y_1用Euler方法确定，而步数N_i总选为偶数（总为奇数也可以，但选偶数有某些优点），则用中点公式计算，总的计算方法具有（10.7.1）的展开式的性质，其中$\gamma=2$。然而这样计算稳定性（参看10.9）是不好的。Gragg在计算过程的最后对$x=x_0+H$作了一些光滑处理，控制了不稳定性而不破坏性质

(10.7.1)．

10.7.8 Gragg（格拉格）外推方法/（Gragg Extrapolation Method）

$h_i = H/N_i$，N_i为偶数

$y_0 = y(x_0)$

$y_1 = y_0 + h_i f(x_0, y_0)$

$y_{n+2} = y_n + 2hf(x_{n+1}, y_{n+1}) \quad (n = 0, 1, 2, \cdots, N_i - 1)$

$$y(x_0 + H; h_i) = \frac{1}{4} y_{N_i - 1} + \frac{1}{2} y_{N_i} + \frac{1}{4} y_{N_i + 1}$$

利用Gragg方法计算，即可用（10.7.3）及（10.7.4）进行外推．其中的N_i，可以取为{2,4,8,16,32,…}，但由于每步都加倍计算量，费时较多，故一般N_i的序列可取为{2,4,6,8,12,16,24,32,…}．

10.7.9 例 用Gragg方法及（10.7.3），（10.7.4）的外推法解

$$\frac{dy}{dx} = 1 - 2xy, \quad 0 \leqslant x \leqslant 1$$

$y(0) = 0$

x_n	不外推		一次外推误差	二次外推误差
	y_n	误差		
0.0	0.00000000	0	0	0
0.1	0.09925250	8.3×10^{-5}	4.1×10^{-8}	2.0×10^{-11}
0.2	0.19458429	1.7×10^{-4}	7.1×10^{-8}	3.8×10^{-11}
0.3	0.28238289	2.5×10^{-4}	7.2×10^{-8}	5.5×10^{-11}
0.4	0.35961606	3.3×10^{-4}	3.2×10^{-8}	7.6×10^{-11}
0.5	0.42403720	4.0×10^{-4}	6.1×10^{-8}	1.1×10^{-10}
0.6	0.47430366	4.6×10^{-4}	2.1×10^{-7}	1.7×10^{-10}
0.7	0.50999998	5.0×10^{-4}	4.1×10^{-7}	2.8×10^{-10}
0.8	0.53157328	5.3×10^{-4}	6.6×10^{-7}	4.6×10^{-10}
0.9	0.54019424	5.3×10^{-4}	9.2×10^{-7}	7.1×10^{-10}
1.0	0.53757097	5.1×10^{-4}	1.2×10^{-6}	1.0×10^{-9}

解 取 $H=0.1$作为基本步，每基本步用Gragg方法及外推过程计算，不外推情形实际上步长为0.05，而外推情形只列出误差，如前表所示。

10.8

10.8 方程组和高阶方程的数值方法

10.8.1 一阶微分方程组的数值方法

对于如下一阶微分方程组的初值问题

10.8.1
$$\begin{cases} \dfrac{dy_1}{dx}=f_1(x,y_1,y_2,\cdots,y_m), & y_1(x_0)=y_{10} \\ \dfrac{dy_2}{dx}=f_2(x,y_1,y_2,\cdots,y_m), & y_2(x_0)=y_{20} \\ \quad\vdots & \quad\vdots \\ \dfrac{dy_m}{dx}=f_m(x,y_1,y_2,\cdots,y_m), & y_m(x_0)=y_{m0} \end{cases}$$

求解的数值方法可以模拟单个方程的某些数值方法来构造。

10.8.2 一阶微分方程组的经典Runge-Kutta方法

$$y_{i,n+1}=y_{i,n}+\frac{h}{6}[k_{1,i}+2k_{2,i}+2k_{3,i}+k_{4,i}]$$

$$(i=1,2,\cdots,m)$$

其中

$$k_{1,i}=f_i(x_n,\ y_{1,n},\ y_{2,n},\cdots,y_{m,n})$$

$$k_{2,i}=f_i(x_n+\frac{h}{2},y_{1,n}+\frac{h}{2}k_{1,1},\ y_{2,n}$$

$$+\frac{h}{2}k_{1,2},\cdots,\ y_{m,n}+\frac{h}{2}k_{1,m})$$

$$k_{3,i}=f_i(x_n+\frac{h}{2},y_{1,n}+\frac{h}{2}k_{2,1},\ y_{2,n}+\frac{h}{2}k_{2,2},\cdots,$$

$$y_{m,n}+\frac{h}{2}k_{2,m})$$

$$k_{4,i}=f_i(x_n+h,y_{1,n}+hk_{3,1},y_{2,n}+hk_{3,2},\cdots,y_{m,n}+hk_{3,m})$$
$$(i=1,2,\cdots,m)$$

求出的$y_{i,n}$是$y_i(x_n)$ 的近似值.

10.8.2 高阶方程的数值方法

高阶方程的初值问题

10.8.3 $$\frac{d^n y}{dx^n}=f(x,y,\frac{dy}{dx},\cdots,\frac{d^{n-1}y}{dx^{n-1}})$$

$$y(x_0)=y_0\ ,\quad \frac{dy(x_0)}{dx}=y_0^{(1)},\cdots,\ \frac{d^{n-1}y(x_0)}{dx^{n-1}}=y_0^{(n-1)}$$

可以化成方程组

10.8.4 $$\begin{cases}\dfrac{dy_1}{dx}=y_2\ , & y_1(x_0)=y_0\\ \vdots & \vdots\\ \dfrac{dy_{n-1}}{dx}=y_n\ , & y_{n-1}(x_0)=y_0^{(n-2)}\\ \dfrac{dy_n}{dx}=f(x,y_1,y_2,\cdots,y_n), & y_n(x_0)=y_0^{(n-1)}\end{cases}$$

其中 $y=y_1$.可以用解方程组的方法求解.

10.8.5 例 用经典Runge-Kutta方法解初值问题

$$\frac{d^2y}{dx^2}-2\frac{dy}{dx}+2y=e^{2x}\sin x,\quad 0\leqslant x\leqslant 1$$

$$y(0)=-0.4\ ,\quad \frac{dy(0)}{dx}=-0.6$$

解 令 $y_1(x)=y(x)$，$y_2(x)=\dfrac{dy}{dx}$，则得到方程组

$$\begin{cases} \dfrac{dy_1}{dx}=y_2(x) \\ \dfrac{dy_2}{dx}=e^{2x}\sin x-2y_1+2y_2 \end{cases}$$

初始条件为 $y_1(0)=-0.4$，$y_2(0)=-0.6$。按(10.8.2)的方法求解，并与准确解 $0.2e^{2x}(\sin x-2\cos x)$ 比较，有下表：

x_n	$y_{1,n}$	$y_{2,n}$	$\lvert y(x_n)-y_{1,n}\rvert$
0.0	−0.40000000	−0.60000000	0
0.1	−0.46173334	−0.63163124	3.7×10^{-7}
0.2	−0.52555988	−0.64014895	8.3×10^{-7}
0.3	−0.58860144	−0.61366381	1.39×10^{-6}
0.4	−0.64661231	−0.53658203	2.03×10^{-6}
0.5	−0.69356666	−0.38873810	2.71×10^{-6}
0.6	−0.72115190	−0.14438087	3.41×10^{-6}
0.7	−0.71815295	0.22899702	4.06×10^{-6}
0.8	−0.66971133	0.77199180	4.56×10^{-6}
0.9	−0.56644290	0.15347815	4.76×10^{-6}
1.0	−0.35339886	0.25787663	4.50×10^{-6}

10.9

10.9 稳 定 性

使用某一数值方法解初值问题时，计算中总会出现误差，其计算误差的传播情况如何？即步长 h 取定以后，随着步数的增加，误差会不会积累到超过所许可的范围？这就是稳定性的问题。

10.9.1 单步法的绝对稳定性

10.9.1 定义/绝对稳定性/Absolute Stability 对于初值问题（10.1.1），（10.1.2），用某一单步方法，以固定步长 h 进行计算．若在一个节点上值 y_n 有扰动 δ，由此引起以后各节点上值 y_m（$m>n$）的变化不超过 δ，就称此单步方法是绝对稳定的．

讨论一个方法的稳定性，常常用"试验方程"的方法，即考虑方程

10.9.2 $\dfrac{dy}{dx}=\lambda y$

10.9.3 $y(x_0)=y_0$

其中 λ 一般可以取复数值．并设 $\mathrm{Re}\lambda<0$，这样能保证微分方程（10.9.2）是稳定的．用这样简单的方程来讨论稳定性，是因为一个数值方法对解（10.9.2）这样简单的方程若是不稳定，则对更复杂的方程也未必稳定．同时，若（10.1.1）中右端 $f(x,y)$ 展开为

$$f(x,y)=f(a,b)+(x-a)\frac{\partial f}{\partial x}(a,b)+(y-b)\frac{\partial f(a,b)}{\partial y}+\cdots$$

在略去高阶项后，可作变量置换化为（10.9.2）．

如果用一个单步方法解（10.9.2），可以得出

10.9.4 $y_{n+1}=r_1(\lambda h)y_n$

在 y_0 上若有扰动 δ_0，并设以后计算是精确的，则在 y_{n+1} 引起误差为

$$\delta_{n+1}=r_1(\lambda h)\delta_n=(r_1(\lambda h))^{n+1}\delta_0$$

所以，若记 $\mu=\lambda h$，如要求方法是绝对稳定的，就要求

10.9.5 $|r_1(\mu)|\leqslant 1$

10.9.6 定义/绝对稳定区域/Region of Absolute Stability

复平面μ中，使$|\gamma_1(\mu)|<1$成立的区域，称为对应的数值方法的**绝对稳定区域**。

10.9.7 定义/绝对稳定区间/Interval of Absolute Stability

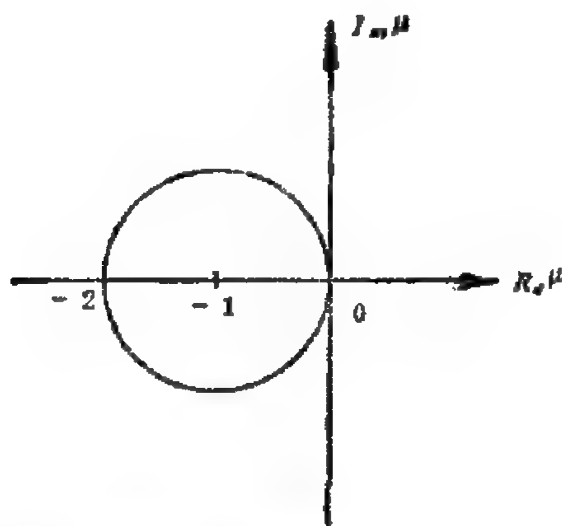
绝对稳定区域与实轴的交称为对应数值方法的**绝对稳定区间**。

10.9.8 例/Euler方法的绝对稳定性 对于方程（10.9.2），Euler方法可写成

$$y_{n+1}=y_n+\lambda h y_n=(1+\mu)y_n$$

所以，当$|1+\mu|\leqslant 1$时，Euler方法是绝对稳定的。

10.9.9 图 Euler方法的绝对稳定区域



Euler方法的绝对稳定区域是一个以（-1，0）为中心的单位圆，它的绝对稳定区间是（-2,0）。所以，当λ为负实数时，取$h\leqslant -\dfrac{2}{\lambda}$，Euler方法是稳定的。

10.9.10 例 用Euler方法计算

$$\frac{dy}{dx}=-1000(y-x^2)+2x$$

$$y(0)=0$$

解 设计算机有10位有效数字的精确度，取$h=10^{-m}$，

$m=0, 1, 2, \cdots$，计算到 $y(1)$，结果如下：

h	$y(1)$
1	0
0.1	$0.90438207503\times10^{20}$
0.01	溢　　出
0.001	0.99999900001
0.0001	0.99999990000
0.00001	0.99999999997

表中“溢出”表示超过了机器能容纳的最大数（本例为 10^{38}）。本例相当 $\lambda=-1000$，当 $h=0.01$ 时，$1+\lambda h=-9$，即每步误差放大 9 倍，计算到 $y(1)$ 有 100 步，第一步的误差约放大 10^{100} 倍，显然这样计算是不合理的。但是，只要 h 适当小，使 $|1+\lambda h|$ 小于 1，计算就合理。

10.9.11　例/隐式Euler方法的绝对稳定性　　把隐式Euler方法（10.3.5）用于方程（10.9.2），得到

$$y_{n+1}=\frac{1}{1-h\lambda}y_n$$

对于实数的 λh，绝对稳定区间为 $(-\infty, 0)$。

10.9.12　例/经典 Runge-Kutta 方法的绝对稳定性　　用经典 Runge-Kutta 方法解（10.9.2），可得到

$$y_{n+1}=\left(1+\lambda h+\frac{(\lambda h)^2}{2}+\frac{(\lambda h)^3}{6}+\frac{(\lambda h)^4}{24}\right)y_n$$

令 $\mu=\lambda h$，则经典 Runge-Kutta 方法的绝对稳定区域为

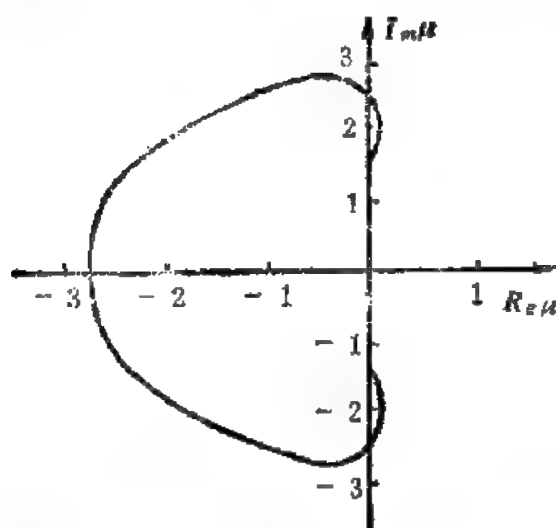
$$\left|1+\mu+\frac{\mu^2}{2}+\frac{\mu^3}{6}+\frac{\mu^4}{24}\right|\leqslant1$$

即在 μ 平面上，由曲线

$$1+\mu+\frac{\mu^2}{2}+\frac{\mu^3}{6}+\frac{\mu^4}{24}=e^{i\theta}$$

所围成的区域。

10.9.13 图　经典Runge-Kutta方法的绝对稳定区域



当λ取实数时，就可得到下述方法的绝对稳定区间。

10.9.14 表　*R*级*R*阶Runge-Kutta方法的绝对稳定区间

R	$r_1(\mu)$	绝对稳定区间
1	$1+\mu$	$(-2,0)$
2	$1+\mu+\frac{\mu^2}{2}$	$(-2,0)$
3	$1+\mu+\frac{\mu^2}{2}+\frac{\mu^3}{6}$	$(-2.51,0)$
4	$1+\mu+\frac{\mu^2}{2}+\frac{\mu^3}{6}+\frac{\mu^4}{24}$	$(-2.78,0)$

10.9.15 例　用经典 Runge-Kutta方法计算

$$\frac{dy}{dx}=-20y,\quad 0\leqslant x\leqslant 1$$

$$y(0)=1$$

取 $h=0.1$ 及0.2。

解　本例 $\lambda=-20$，$\mu=\lambda h$ 分别为 -2和 -4。前者属绝

对稳定区间，后者则不属于，计算结果误差如下表

x_n	$h=0.1$时误差	$h=0.2$时误差
0.0	0	0
0.2	−0.092795	4.98
0.4	−0.012010	25.0
0.6	−0.001366	125.0
0.8	−0.000152	625.0
1.0	−0.000017	3125.0

10.9.2 线性多步法的绝对稳定性

如果给定一个解初值问题(10.1.1)，(10.1.2)的线性多步法（参见10.5）

10.9.16 $$\sum_{j=0}^{k} \alpha_j y_{n+j} = h \sum_{j=0}^{k} \beta_j f_{n+j}$$

则可对应两个以 ξ 为变量的多项式

10.9.17 $$\rho(\xi) = \sum_{j=0}^{k} \alpha_j \xi^j, \quad \sigma(\xi) = \sum_{j=0}^{k} \beta_j \xi^j$$

将(10.9.16)用于试验方程(10.9.2)，可得到一个k阶的常系数线性差分方程

10.9.18 $$\sum_{j=0}^{k} (\alpha_j - \mu\beta_j) y_{n+j} = 0$$

它的特征方程是

10.9.19 $$\rho(\xi) - \mu\sigma(\xi) = 0$$

其中 $\rho(\xi)$，$\sigma(\xi)$ 如（10.9.17）所示，$\mu = \lambda h$。如果(10.9.19)的根是$\xi_j (j=1,2,\cdots,k)$，且都是单根，根据线性差分方程的理论，(10.9.18)的解为

10.9.20 $y_n = \sum_{j=1}^{k} B_j \xi_j^n$

若有一个根 ξ_j 是q重的，则和式中会产生$[B_{j1}+nB_{j2}+n(n-1)B_{j3}+\cdots+n(n-1)\cdots(n-q+2)B_{jq}]\xi_j^n$的项。显然，用多步法（10.9.16）解（10.9.2），y_n 的误差满足差分方程（10.9.18）。若$|\xi_j|<1$，当 n 增大时，由(10.9.20)知误差是有界的，但对 $|\xi_j|=1$，在重根的情况下误差会变成无界的，所以不考虑等号的情形。

10.9.21 **定义/绝对稳定性** 对于给定的μ，若特征方程(10.9.19)的ξ_j满足 $|\xi_j|<1(j=1,2,\cdots,k)$，则称线性多步法(10.9.16)是**绝对稳定的**。

10.9.22 **定义/绝对稳定区域** 复平面μ中，使(10.9.16)绝对稳定的区域，称为(10.9.16)的**绝对稳定区域**。

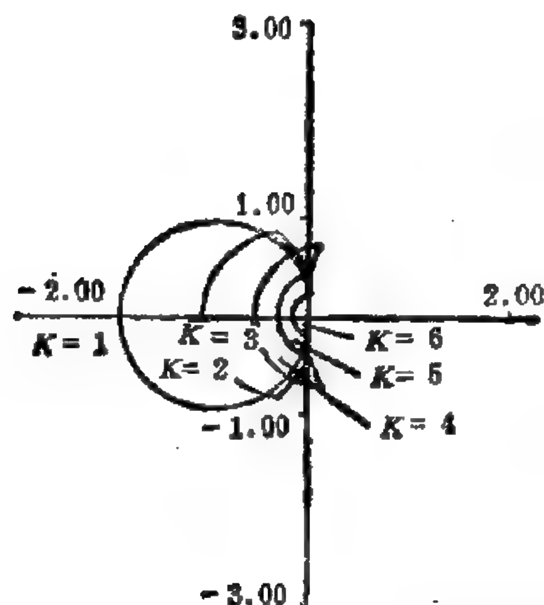
10.9.23 **定义/绝对稳定区间** （10.9.16）的绝对稳定区域与实轴的交称(10.9.16)的**绝对稳定区间**。

10.9.24 **例**

k 步 Adams-Bashforth方法的绝对稳定区间如下

步数k	阶	绝对稳定区间
1	1	$(-2,0)$
2	2	$(-1,0)$
3	3	$(-\frac{6}{11},0)$
4	4	$(-\frac{3}{10},0)$

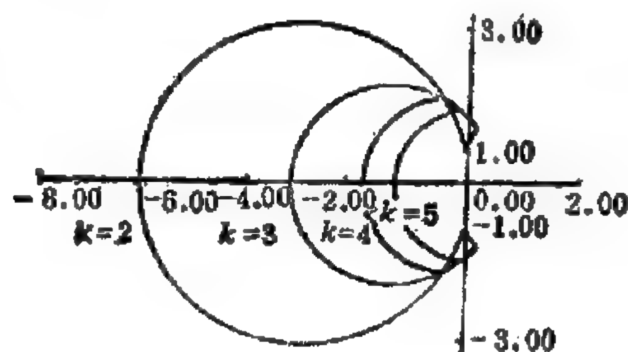
10.9.25 **图** **Adams-Bashforth k步方法的绝对稳定区域**



10.9.26 例 k 步 Adams－Moulton 方法绝对稳定区间如下

步数k	阶	绝对稳定区间
1	2	$(-\infty,0)$
2	3	$(-6,0)$
3	4	$(-3,0)$
4	5	$(-\frac{90}{49},0)$

10.9.27 图 Adams-Moulton k步方法的绝对稳定区域



对于 Milne 方法，可以证明，对于任何 μ，它都不是绝对稳定的。

10.9.28 例 用 Milne 方法解

$$\frac{dy}{dx}=-y$$

$$y(1)=1$$

解 取 $h=10^{-l}$ $(l=1,2,3,4,5)$。计算 $y(10)$，结果如下：

h	$y(10)$	误 差
0.1	0.48625×10^{-4}	-0.32251×10^{-5}
0.01	0.48211×10^{-4}	-0.28114×10^{-5}
0.001	0.38639×10^{-4}	0.67607×10^{-5}
0.0001	0.32934×10^{-4}	-0.37534×10^{-4}
0.00001	0.57846×10^{-4}	-0.12446×10^{-4}

可见，$y(10)$的值和误差几乎有相同的数量级。

Milne 方法的稳定性不好，而 Hamming 方法及由它组成的预测—校正方法，则有较好的稳定性。

10.9.3 方程组线性多步法的绝对稳定性

将m个方程的方程组初值问题(10.8.1)写成向量形式

10.9.29 $$\frac{d\boldsymbol{y}}{dx}=\boldsymbol{f}(x,\boldsymbol{y})$$

10.9.30 $$\boldsymbol{y}(0)=\boldsymbol{y}_0$$

类似一个方程情形，线性多步法为

10.9.31 $$\sum_{j=0}^{k}\alpha_j\boldsymbol{y}_{n+j}=h\sum_{j=0}^{k}\beta_j\boldsymbol{f}_{n+j}$$

讨论(10.9.31)误差增长的情况，可以考虑线性常系数方程

$$\frac{dy}{dx}=Jy+f(x)$$

其中 J 为 $m\times m$的常数矩阵，将(10.9.31) 用于此方程组，其误差增长情况取决于特征方程 $\rho(\xi)-\mu\sigma(\xi)=0$ 的根，这里 $\rho(\xi)$,$\sigma(\xi)$同(10.9.17)，$\mu=\lambda h$，λ 取遍 J 的特征值。

10.9.32 **定义/绝对稳定性** 对于解方程组的多步法(10.9.31)，如给定 μ，方程 $\rho(\xi)-\mu\sigma(\xi)=0$ 的根 $|\xi_j|<1(j=1,2,\cdots,k)$，则称(10.9.31)绝对稳定。

同样可以类似(10.9.22)定义(10.9.31)的绝对稳定区域。

10.10

10.10 刚性方程组的数值方法

10.10.1 方程组的刚性现象

用一种数值方法解方程组，为了保证绝对稳定性，选取步长 h 使得 λh 属于绝对稳定区域，其中 λ 应取遍系数矩阵的特征值，但是在这些特征值相差十分悬殊时，计算就会有很大困难。

10.10.1 **例** 关于方程组

$$\begin{cases} y_1'=-1001y_1+999y_2+2, & y_1(0)=3 \\ y_2'=999y_1-1001y_2+2, & y_2(0)=1 \end{cases}$$

其系数矩阵为$\begin{bmatrix}-1001 & 999\\ 999 & -1001\end{bmatrix}$，特征值为 $\lambda_1=-2000$，$\lambda_2=-2$，方程组的解为

$$y_1(x)=e^{-2000x}+e^{-2x}+1$$

$$y_2(x)=-e^{-2000x}+e^{-2x}+1$$

当$x\to\infty$时，$y_1(x)$和$y_2(x)$都趋于稳定值1，然而它们包含

的两项e^{-2000x}和e^{-2x}性质却有很大差别。前者“瞬变”过程短，约在$x=0.01$就接近稳定值；后者变化慢，约在$x=10$才接近稳定值。如果用经典四阶 Runge-Kutta 方法求解，为了将“瞬变”部分表现出来，在$0\leqslant x\leqslant 0.01$应该取小步长，同时，为了达到绝对稳定，应取$\lambda h\in(-2.78,0)$。由于$\lambda_1=-2000$，所以$h<0.00139$，而到达$x=0.01$后，快速瞬变部分已趋稳定，则希望取较大的步长。但是，为了保证稳定性，仍要取小的h值，这样到$x=10$至少要7200步，给计算带来很大困难，所以这类方程组要用特殊的方法。

10.10.2 定义/刚性方程组/Stiff Systems 设方程组

$$\frac{dy}{dx}=Jy+f(x)$$

系数矩阵J的特征值为$\lambda_j(j=1,2,\cdots,m)$。若

1° $\mathrm{Re}\lambda_j<0 \quad (j=1,2,\cdots,m)$

2° $$S=\frac{\max\limits_j|\mathrm{Re}\lambda_j|}{\min\limits_j|\mathrm{Re}\lambda_j|}>>1$$

则称方程组是刚性的，S称为刚性比（Stiffness Ratio）。

对于一般的非线性方程组(10.9.29)，在以λ_j表示Jocobi矩阵$\frac{\partial f}{\partial y}$的特征值后，若满足(10.10.2)中的1°，2°，同样定义为刚性的方程组。

在化学工程，控制理论，网络等领域的微分方程组中，往往出现刚性的方程组。实际问题的刚性比S，有时可以取到10^{10}或更大的值。

10.10.2 刚性方程组的数值方法

设有方程组

10.10.3 $\frac{dy}{dx}=f(x,y)$

10.10.4 Gear（吉尔）方法/Gear Method　Gaer 方法是如下的隐式多步方法，

$$\sum_{j=0}^{k} \alpha_j y_{n+j}=h\beta_k f_{n+k}$$

其中系数如下表：

k	β_k	α_0	α_1	α_2	α_3	α_4	α_5	α_6
1	1	-1	1					
2	$\frac{2}{3}$	$\frac{1}{3}$	$-\frac{3}{4}$	1				
3	$\frac{6}{11}$	$-\frac{2}{11}$	$\frac{9}{11}$	$-\frac{18}{11}$	1			
4	$\frac{12}{25}$	$\frac{3}{25}$	$\frac{-16}{25}$	$\frac{36}{25}$	$-\frac{48}{25}$	1		
5	$\frac{60}{137}$	$-\frac{12}{137}$	$\frac{75}{137}$	$-\frac{200}{137}$	$\frac{300}{137}$	$-\frac{300}{137}$	1	
6	$\frac{60}{147}$	$\frac{10}{147}$	$-\frac{72}{147}$	$\frac{225}{147}$	$-\frac{400}{147}$	$\frac{450}{147}$	$-\frac{360}{147}$	1

k 步的 Gear 方法是 k 阶的．$k=1$时就是隐式的Euler方法．用 Gear 方法计算时，要解关于 y_{n+k} 的非线性方程组，一般用 Newton 方法求解，而不用类似(10.1.21)的简单迭代法．

解刚性方程组可用隐式的Runge-Kutta方法，如使用

10.10.5 Hammer-Hollingsworth（哈默-荷令斯沃思） 二级四阶方法

$$y_{n+1}=y_n+\frac{h}{2}(k_1+k_2)$$

其中

$$k_1=f\Big(x_n+(\frac{1}{2}+\frac{\sqrt{3}}{6})h,\ y_n+\frac{1}{4}hk_1+(\frac{1}{4}+\frac{\sqrt{3}}{6})hk_2\Big)$$

$$k_2=f\Big(x_n+(\frac{1}{2}-\frac{\sqrt{3}}{6})h,\ y_n+(\frac{1}{4}-\frac{\sqrt{3}}{6})hk_1+\frac{1}{4}hk_2\Big)$$

(10.10.5)只是单个方程的计算方法，它的绝对稳定区间是$(-\infty,0)$，可以把它推广到方程组的情形。如果用R级的隐式 Runge-Kutta 方法解m个方程的刚性方程组，虽然稳定性质较好，但每一步要解mR个非线性方程的方程组，且用 Newton 法求解要有较好的初值估计，故实际上并不容易。

10.10.6 Bui(布意)四级四阶方法 对于方程

$$\frac{d\boldsymbol{y}}{dx}=\boldsymbol{f}(\boldsymbol{y})$$

有

$$\boldsymbol{y}_{n+1}=\boldsymbol{y}_n+\sum_{r=1}^{4}c_r\boldsymbol{K}_r$$

式中$\boldsymbol{K}_r$由方程

$$\Big[I-\gamma h\frac{\partial \boldsymbol{f}(\boldsymbol{y}_n)}{\partial \boldsymbol{y}}\Big]\boldsymbol{K}_r=h\boldsymbol{f}(\boldsymbol{y}_n+\sum_{s=1}^{r-1}b_{rs}\boldsymbol{K}_s),\quad (r=1,2,3,4)$$

决定，其中$\frac{\partial \boldsymbol{f}}{\partial \boldsymbol{y}}$为 Jacobi 矩阵。求解$\boldsymbol{K}_r$只有四个$m$个方程

的线性方程组，且四个方程组有相同的系数矩阵，所以每步只要作一次矩阵分解。

Bui 方法的系数是：

$r=0.5728160625$

$b_{21}=-0.5$

$b_{31}=-0.1012236115$，$b_{32}=0.9762236115$

$b_{41}=-0.3922096763$，$b_{42}=0.7151140251$

$b_{43}=0.1430371625$

$c_1=0.9451564786$，$c_2=0.341323172$

$c_3=0.5655139575$，$c_4=-0.8519936081$

11 # 第11章　常微分方程边值问题的数值方法

11.1 ## 11.1 引　言

常微分方程边值问题的数值方法的典型例子是二阶微分方程的两点边值问题**Two-Point Boundary Value Problem**，它的一般形式是

11.1.1　$y''=f(x,y,y'),\quad a\leqslant x\leqslant b$

11.1.2　$y(a)=\alpha,\quad y(b)=\beta$

其中(11.1.1)是$[a,b]$上一个二阶常微分方程.(11.1.2)是在边界点a,b上的条件，它给出了a,b上的函数值．通常，边界条件还有其它的给法，例如

11.1.3　$\alpha_1 y(a)+\beta_1 y'(a)=\alpha,\quad \alpha_2 y(b)+\beta_2 y'(b)=\beta$

其中$|\alpha_1|+|\beta_1|\neq 0$，$|\alpha_2|+|\beta_2|\neq 0$.

(11.1.3)第一式中，若$\alpha_1\neq 0$，$\beta_1\neq 0$，称为第三类边界条件．若$\alpha_1=0$，称为第二类边界条件．若$\beta_1=0$，则化为(11.1.2)的形式，称第一类边界条件．(11.1.3)的第二式与第一式类同.(11.1.3)和(11.1.2)都是线性的边界条件，在很多实际问题中还需考虑非线性的边界条件．

11.1.4　定理　设边界问题(11.1.1),(11.1.2)中,函数f及$\frac{\partial f}{\partial y}$，$\frac{\partial f}{\partial y'}$在区域

$$D=\{(x,y,y')\,|a\leqslant x\leqslant b,-\infty<y<+\infty,\\ -\infty<y'<+\infty\}$$

连续，且

1° $\frac{\partial f}{\partial y}(x,y,y')>0$, $\forall(x,y,y')\in D$

2° 存在常数 $M>0$，使得

$$\left|\frac{\partial f}{\partial y'}(x,y,y')\right|\leqslant M, \quad \forall(x,y,y')\in D$$

则边界问题（11.1.1），（11.1.2）有唯一解。

11.1.5 **定理** 对于线性的边值问题

11.1.6 $y''=p(x)y'+q(x)y+r(x)$, $a\leqslant x\leqslant b$

$y(a)=\alpha$, $y(b)=\beta$

若满足

1° p, q, r 在$[a, b]$上连续；

2° $q(x)>0$, $\forall x\in[a,b]$

则（11.1.6），（11.1.2）有唯一解。

对应于（11.1.6）的边值问题，可以考虑两个初值问题：

11.1.7 $\begin{cases} y_1''=p(x)y_1'+q(x)y_1+r(x), & a\leqslant x\leqslant b \end{cases}$

11.1.8 $\quad y_1(a)=\alpha$, $y_1'(a)=0$

及

11.1.9 $\begin{cases} y_2''=p(x)y_2'+q(x)y_2, & a\leqslant x\leqslant b \end{cases}$

11.1.10 $\quad y_2(a)=0$, $y_2'(a)=1$

11.1.11 **定理** 设 y_1是初值问题（11.1.7），（11.1.8）的解，y_2 是初值问题（11.1.9），（11.1.10）的解，并设$y_2(b)\neq 0$，则边值问题（11.7.6），（11.1.2）的解为

11.1.12 $$y(x)=y_1(x)+\frac{\beta-y_1(b)}{y_2(b)}y_2(x)$$

11.2

11.2 试 射 法

11.2.1 线性边值问题的试射法

为了求解线性边值问题（11.1.6），（11.1.2），在运用定理（11.1.11）后，可得到

11.2.1 线性边值问题的试射法 用解初值问题的数值方法（例如经典的 Runge-Kutta方法，见10.4.11）分别解(11.1.7)，(11.1.8)和(11.1.9)，(11.1.10)，得到$y_1(x)$和$y_2(x)$，然后利用(11.1.12)，得到边值问题(11.1.6)，(11.1.2) 的数值解。

11.2.2 定理 在方法(11.2.1) 中，若 y_{1n} 及 y_{2n} 分别为$y_1(x)$及 $y_2(x)$在节点 x_n 函数值的 $O(h^p)$近似值($n=0,1,\cdots,N$)，则 y_n 亦为 $y(x_n)$的 $O(h^p)$近似值，且存在常数 $K>0$，使得

$$|y(x_n)-y_n|\leqslant Kh^p\left|1+\frac{y_{2n}}{y_{2N}}\right|$$

11.2.3 例 用试射法解线性边值问题

$$y''=-\frac{2}{x}y'+\frac{2}{x^2}y+\frac{\sin(\ln x)}{x^2},\quad 1\leqslant x\leqslant 2$$

$$y(1)=1,\quad y(2)=2$$

解 根据定理（11.1.12），本例求解边值问题可以化为解两个初值问题

$$\begin{cases} y_1''=-\dfrac{2}{x}y_1'+\dfrac{2}{x^2}y_1+\dfrac{\sin(\ln x)}{x^2},\quad 1\leqslant x\leqslant 2 \\ y_1(1)=1,\quad y_1'(1)=0 \end{cases}$$

$$\begin{cases} y_2''=-\dfrac{2}{x}y_2'+\dfrac{2}{x^2}y_2,\quad 1\leqslant x\leqslant 2 \\ y_2(1)=0,\quad y_2'(1)=1 \end{cases}$$

分别用经典 Runge-Kutta 方法求解，取 $h=0.1$，得到y_{1n}，y_{2n}，再由 $y_{1n}+\dfrac{2-y_{1N}}{y_{2N}}y_{2n}$ 计算 y_n，计算结果如下：

x_n	y_{1n}	y_{2n}	y_n	准确解 $y(x_n)$	$\lvert y(x_n) - y_n \rvert$
1.0	1.00000000	0.00000000	1.00000000	1.00000000	—
1.1	1.00896058	0.09117986	1.09262917	1.09262930	1.43×10^{-7}
1.2	1.03245472	0.16851175	1.18708471	1.18708484	1.34×10^{-7}
1.3	1.06674375	0.23608704	1.28338227	1.28338236	9.78×10^{-8}
1.4	1.10928795	0.29659067	1.38144589	1.38144595	6.02×10^{-8}
1.5	1.15830000	0.35184379	1.48115939	1.48115942	3.06×10^{-8}
1.6	1.21248372	0.40311695	1.58239245	1.58239246	1.08×10^{-8}
1.7	1.27087454	0.45131840	1.68501396	1.68501396	5.43×10^{-10}
1.8	1.33273851	0.49711137	1.78889854	1.78889853	5.05×10^{-9}
1.9	1.39750618	0.54098928	1.89392951	1.89392951	4.41×10^{-9}
2.0	1.46472815	0.58332538	2.00000000	2.00000000	—

例（11.2.3）所用的方法有比较好的效果，但有时由于误差的作用，在计算（11.1.12）时，会引起有效数字的消失得不到满意的结果。因此，有时可以用相反方向的试射法解边值问题，即分别解初值问题

$$\begin{cases} y_1'' = p(x)y_1' + q(x)y_1 + r(x), & a \leqslant x \leqslant b \\ y_1(b) = \beta, \quad y_1'(b) = 0 \end{cases}$$

$$\begin{cases} y_2'' = p(x)y_2' + q(x)y_2 \\ y_2(b) = 0, \quad y_2'(b) = 1 \end{cases}$$

设它们的解分别为 y_1 和 y_2，类似（11.1.12）有

$$y(x) = y_1(x) + \frac{\alpha - y_1(a)}{y_2(a)} y_2(x)$$

11.2.2 非线性问题的试射法

在二阶方程的边值问题（11.1.1），（11.1.2）是非线性的情形下，若有参数 t，初值问题

11.2.4 $\quad y'' = f(x, y, y'), \quad a \leqslant x \leqslant b$

11.2.5 $y(a)=\alpha$, $y'(a)=t$

的解记为 y，其函数值与 x、t有关．如果此初值问题的解与边值问题（11.1.1），（11.1.2）的解相同，即

11.2.6 $y(b,t)=\beta$

则可用解初值问题代替解边值问题．但是 t 并不是很容易确定的，故用一系列$\{t_k\}$来逼近它．

11.2.7 **定理** 设在区域

$$D=\{(x,y,y')\,|a\leqslant x\leqslant b,-\infty<y<+\infty,\\ -\infty<y'<+\infty\}$$

上，$f(x,y,y')$满足：

1° $f,\dfrac{\partial f}{\partial y},\dfrac{\partial f}{\partial y'}$连续；

2° 存在常数 $M>0$，使 $\left|\dfrac{\partial f}{\partial y'}\right|\leqslant M$；

3° 存在常数 $L>0$，使$0<\dfrac{\partial f}{\partial y}<L$；

则边值问题（11.1.1），（11.1.2）及初值问题（11.2.4），（11.2.5）对于任意的参数 t，均在$[a,b]$上存在唯一的解．

11.2.8 **试射法** 用数值方法解初值问题

11.2.9 $y''=f(x,y,y')$，$a\leqslant x\leqslant b$

11.2.10 $y(a)=\alpha$，$y'(a)=t_k$

得到解$y(x,t_k)$，若对于给定的误差限 ε，有 $|y(b,t_k)-\beta|<\varepsilon$，则认为 $y(x,t_k)$是边值问题（11.1.1），（11.1.2）的解，否则，将 t_k 修改为 t_{k+1}，重复上述计算．

要想成功地运用试射法，就要作到 $\lim\limits_{k\to\infty}y(b,t_k)=\beta$．实际上，$\{t_k\}$可以看成是非线性方程（11.2.6）的一个近似解序列，所以试射法的具体计算步骤，可以采用解非线性方程

的一些方法（参见第5章）。

11.2.11 二分法/Bisection Method 记

11.2.12 $F(t_k)=y(b,t_k)-\beta$

在（11.2.8）中，选择初值 t_1，t_2，使 $F(t_1)\cdot F(t_2)<0$。一般地，若已知 $t_k<t_{k+1}$，可构成迭代区间 $[t_k,t_{k+1}]$。

设 $F(t_k)\cdot F(t_{k+1})<0$，令

11.2.13 $t_{k+2}=\dfrac{1}{2}(t_k+t_{k+1})$

将 $[t_k, t_{k+1}]$ 分半，若 $F(t_{k+2})\cdot F(t_k)<0$，则以 $[t_k,t_{k+2}]$ 代替 $[t_k,t_{k+1}]$，否则以 $[t_{k+2}, t_{k+1}]$ 代替 $[t_k, t_{k+1}]$，继续进行迭代。

11.2.14 割线法/Secant Method 在（11.2.8）中，选定初值 t_1,t_2，一般地，若已知 t_k，t_{k+1}，则

$$t_{k+2}=t_{k+1}-\frac{[y(b,t_{k+1})-\beta](t_{k+1}-t_k)}{y(b,t_{k+1})-y(b,t_k)}$$

Newton 法（参见第5章）是解非线性方程的一种有效方法。把它用于（11.2.8），即解(11.2.6)，要用到 $\dfrac{\partial y(b,t)}{\partial t}$ 之值，而 $y(b,t)$ 的解的表达式是未知的，若把 $y(x,t)$ 看成是初值问题（11.2.4），（11.2.5）的解，则两式分别对 t 求导，有：

$$\frac{\partial y''}{\partial t}=\frac{\partial f}{\partial y}\cdot\frac{\partial y}{\partial t}+\frac{\partial f}{\partial y'}\cdot\frac{\partial y'}{\partial t}$$

$$\frac{\partial y(a,t)}{\partial t}=0,\quad \frac{\partial y'(a,t)}{\partial t}=1$$

以 $z(x,t)$ 记 $\dfrac{\partial y(x,t)}{\partial t}$，并设对 x 和 t 的导数可交换次序。于是可得关于 z 的一个微分方程的初值问题。即（11.2.16），

(11.2.17)，所以 Newton 法用于（11.2.8）为：

11.2.15 **Newton(牛顿)法** 在（11.2.8）中，给定初值 t_1，一般地，若给出 t_k，则从（11.2.9），（11.2.10）解出 $y(x,t_k)$，再从

11.2.16 $z''=\frac{\partial f(x,y,y')}{\partial y}z+\frac{\partial f(x,y,y')}{\partial y'}z'$，$a\leqslant x\leqslant b$

11.2.17 $z(a)=0$，$z'(a)=1$

解出 $z(x,t)$，则

11.2.18 $t_{k+1}=t_k-\frac{y(b,t_k)-\beta}{z(b,t_k)}$

11.2.19 **例** 用试射法的 Newton 迭代过程解边值问题

$$y''=\frac{1}{8}(32+2x^3-yy'),\ 1\leqslant x\leqslant 3$$

$$y(1)=17, y(3)=\frac{43}{3}$$

解 依据 Newton 法，把解边值问题化成每步求解两个初值问题：

$$\begin{cases}y''=\frac{1}{8}(32+2x^3-yy'),\ 1\leqslant x\leqslant 3\\ y(1)=17,\ y'(1)=t_k\end{cases}$$

$$\begin{cases}z''=-\frac{1}{8}(yz'+y'z),\ 1\leqslant x\leqslant 3\\ z(1)=0,\ z'(1)=1\end{cases}$$

给定初值 t_1，根据（11.2.18）进行迭代，直到

$$\left|y(b,t_k)-\frac{43}{3}\right|<\varepsilon$$

时迭代结束.

11.3 有限差分方法

11.3.1 线性问题的差分方法

对于二阶线性方程的两点边值问题

11.3.1 $y''=p(x)y'+q(x)y+r(x),\ a\leqslant x\leqslant b$

11.3.2 $y(a)=\alpha,\ y(b)=\beta$

(11.3.2) 为第一类边界条件，也可以考虑第二、三类边界条件：

11.3.3 $y'(a)=\alpha,\ y'(b)=\beta$

11.3.4 $y'(a)-\alpha_1 y(a)=\alpha,\ y'(b)+\beta_1 y(b)=\beta$

其中 $\alpha_1\geqslant 0,\beta_1\geqslant 0$。对于左端点 $x=a$和右端点 $x=b$，也可以分别取不同类型的边界条件.

在 (11.3.1) 两边同乘$e^{-\int p(x)dx}$,再令 $t=\int e^{\int p(x)dx}dx$,方程化成

$$\frac{dy^2}{dt^2}+Q(t)y=R(t)$$

所以有时考虑 (11.3.1) 中 $p(x)\equiv 0$.

将$[a,b]$区间用节点 $x_i(i=0,1,\cdots,N)$ 分为 N 等分，记 $x_i=a+ih(i=0,1,\cdots,N),h=\dfrac{b-a}{N}$. 由 Taylor 公式，设 $y\in C^4[a,b]$,则有

$$y(x_{i+1})=y(x_i)+hy'(x_i)+\frac{h^2}{2}y''(x_i)$$
$$+\frac{h^3}{6}y'''(x_i)+\frac{h^4}{24}y^{(4)}(\xi_i^+)$$

$$y(x_{i-1})=y(x_i)-hy'(x_i)+\frac{h^2}{2}y''(x_i)$$

$$-\frac{h^3}{6}y'''(x_i)+\frac{h^4}{24}y^{(4)}(\xi_i^-)$$

其中 $x_i<\xi_i^+<x_{i+1}$，$x_{i-1}<\xi_i^-<x_i$，对此两式用中值定理，得到

11.3.5 $$y''(x_i)=\frac{1}{h^2}[y(x_{i+1})-2y(x_i)+y(x_{i-1})]-\frac{h^2}{12}y^{(4)}(\xi_i)$$

其中 $x_{i-1}<\xi_i<x_{i+1}$，$i=1,2,\cdots,n-1$。

同理可得

11.3.6 $$y'(x_i)=\frac{1}{2h}[y(x_{i+1})-y(x_{i-1})]-\frac{h^2}{6}y'''(\eta_i)$$

其中 $x_{i-1}<\eta_i<x_{i+1}$，$i=1, 2, \cdots, n-1$。(11.3.6) 称为 $y'(x_i)$的中心差分公式。

将(11.3.5)，(11.3.6)代入(11.3.1)，得到

$$\frac{y(x_{i+1})-2y(x_i)+y(x_{i-1})}{h^2}=p(x)\left[\frac{y(x_{i+1})-y(x_{i-1})}{2h}\right]+q(x_i)y(x_i)+r(x_i)-\frac{h^2}{12}[2p(x_i)y'''(\eta_i)-y^{(4)}(\xi_i)]$$

将方程的最后一项略去（即截断误差为$O(h^2)$），加上边界条件(11.3.2)，就得到解边值问题(11.3.1)，(11.3.2)的差分方程组：

11.3.7

$$\begin{cases} y_0=\alpha \\ \dfrac{y_{i+1}-2y_i+y_{i-1}}{h^2}-p(x_i)\dfrac{y_{i+1}-y_{i-1}}{2h}-q(x_i)y_i=r(x_i) \\ \qquad\qquad (i=1,2,\cdots,N-1) \\ y_N=\beta \end{cases}$$

写成矩阵形式是

11.3.8 $Ay=b$

其中

$$y=\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_{N-1} \\ y_N \end{pmatrix} \qquad b=\begin{pmatrix} \alpha \\ h^2r(x_1) \\ \vdots \\ h^2r(x_{N-1}) \\ \beta \end{pmatrix}$$

$$A=\begin{pmatrix} 1 & 0 & & & & & 0 \\ 1+\frac{h}{2}p(x_1) & -(2+h^2q(x_1)) & 1-\frac{h}{2}p(x_1) & & & & \\ & 1+\frac{h}{2}p(x_2) & -(2+h^2q(x_2)) & 1-\frac{h}{2}p(x_2) & & & \\ & & \ddots & \ddots & \ddots & & \\ 0 & & & & 1+\frac{h}{2}p(x_{N-1}) & -(2+h^2q(x_{N-1})) & 1-\frac{h}{2}p(x_{N-1}) \\ & & & & & 0 & 1 \end{pmatrix}$$

若解出(11.3.7)的 y_i ($i=0, 1, \cdots, N$)，就求得$y(x_i)$的近似值.

对于第二、三类边界条件(11.3.3)，(11.3.4)，导数值亦要用差分近似.可以用

$$y'(x_0)=\frac{y(x_1)-y(x_0)}{h}+O(h),$$

$$y'(x_N)=\frac{y(x_N)-y(x_{N-1})}{h}+O(h)$$

为了使截断误差达到 $O(h^2)$，可利用 Newton 等距插值公式

$$y'(x_0)=\frac{-y(x_2)+4y(x_1)-3y(x_0)}{2h}+O(h^2)$$

$$y'(x_N)=\frac{3y(x_N)-4y(x_{N-1})+y(x_{N-2})}{2h}+O(h^2)$$

这样，对于第三类边值问题(11.3.1)，(11.3.4)，得到差分方程组

$$\text{11.3.9}\quad\begin{cases}\dfrac{-y_2+4y_1-3y_0}{2h}-\alpha_1 y_0=\alpha \\ \dfrac{y_{i+1}-2y_i+y_{i-1}}{h^2}-p(x_i)\dfrac{y_{i+1}-y_{i-1}}{2h}-q(x_i)y_i \\ \qquad =r(x_i)\qquad (i=1,2,\cdots,N-1) \\ \dfrac{3y_N-4y_{N-1}+y_{N-2}}{2h}+\beta_1 y_N=\beta\end{cases}$$

类似地可得出第二类边值问题的方程组.

11.3.10 定理 设在 $[a, b]$上$q(x)\geqslant 0, L=\max\limits_{a\leqslant b\leqslant x}|p(x)|$，则当 $h<\dfrac{2}{L}$时，方程组(11.3.7)存在唯一的解.

11.3.11 定理 设在 $[a, b]$上 $q(x)\geqslant 0$，$p(x)\equiv 0$，边值问题(11.3.1)，(11.3.2)的解为 y，$y\in C^4[a, b]$，$M_4=\max\limits_{a\leqslant b\leqslant x}|y^{(4)}(x)|$，$y_i$ ($i=0,1, \cdots, N$) 为(11.3.7)的解，则

$$|y(x_i)-y_i|\leqslant\frac{M_4(b-a)^2}{96}h^2$$

定理(11.3.11)说明，若 $x=x_i$ 固定，当 $h\to0$ 时，差分方程的解收敛到微分方程边值问题的解.

差分方程(11.3.7)是一个三对角的线性代数方程组，可以用追赶法来求解（见第 6 章）.

11.3.12 例 用差分方法解线性边值问题

$$y''=-\frac{2}{x}y'+\frac{2}{x^2}y+\frac{\sin(\ln x)}{x^2},\qquad 1\leqslant x\leqslant2$$

$$y(1)=1,y(2)=2$$

解 设 $h=0.1$，$N=10$. 用上述的方法列出差分方程(11.3.7)，然后用追赶法求解，结果如下表：

x_i	y_i	$y(x_i)$	$\lvert y_i-y(x_i)\rvert$
1.0	1.00000000	1.00000000	—
1.1	1.09260052	1.09262930	2.88×10^{-5}
1.2	1.18704313	1.18708484	4.17×10^{-5}
1.3	1.28333687	1.28338236	4.55×10^{-5}
1.4	1.38140205	1.38144595	4.39×10^{-5}
1.5	1.48112026	1.48115942	3.92×10^{-5}
1.6	1.58235990	1.58239246	3.26×10^{-5}
1.7	1.68498902	1.68501396	2.49×10^{-5}
1.8	1.78888175	1.78889853	1.68×10^{-5}
1.9	1.89392110	1.89392951	8.41×10^{-6}
2.0	2.00000000	2.00000000	—

与例(11.2.3) 比较，用差分方法的精确度不如例（11.2.3)，因为在例（11.2.3）中用的是四阶的 Runge-Kutta方法，而这里使用的差分方法的截断误差是 $O(h^2)$. 虽然可以构造截断误差更高阶的差分方程，例如截断误差是 $O(h^4)$，

但由于这样近似 $y''(x_i)$ 时要用到 $y(x_{i-2})$ 和 $y(x_{i+2})$ 的值，而使差分方程更为复杂，故一般不使用。

可以运用类似（10.7）的外推方法，或类似第 4 章中 Romberg 积分算法，对差分方法的结果进行外推。

11.3.13 **例** 用差分方法的外推过程解（11.3.12）的边值问题。

解 设 $h=0.1$，0.05及0.025，作第一次外推为

$$Ext_{1i}=\frac{4y_i(h=0.05)-y_i(h=0.1)}{3}$$

第二次和第三次外推分别为

$$Ext_{2i}=\frac{4y_i(h=0.025)-y_i(h=0.05)}{3}$$

$$Ext_{3i}=\frac{16Ext_{2i}-Ext_{1i}}{15}$$

结果如下表，其中 Ext_{3i} 列出的是经过修正的数字，事实上计算结果与准确解在节点上最大误差为 6.3×10^{-11}。表中略去 $y_i(h=0.1)$ 之值，见例(11.3.12)。

x_i	y_i $(h=0.05)$	y_i $(h=0.025)$	Ext_{1i}	Ext_{2i}	Ext_{3i}
1.0	1.00000000	1.00000000	1.00000000	1.00000000	1.00000000
1.1	1.09262207	1.09262479	1.09262925	1.09262930	1.09262930
1.2	1.18707436	1.18708222	1.18708477	1.18708484	1.18708484
1.3	1.28337094	1.28337950	1.28338230	1.28338236	1.28338236
1.4	1.38143493	1.38144319	1.38144589	1.38144595	1.38144595
1.5	1.48114959	1.48115696	1.48115937	1.48115941	1.48115942
1.6	1.58238429	1.58239042	1.58239242	1.58239246	1.58239246
1.7	1.68500770	1.68501240	1.68501393	1.68501396	1.68501396
1.8	1.78889432	1.78889748	1.78889852	1.78889853	1.78889853
1.9	1.89392740	1.89392898	1.89392950	1.89392951	1.89392951
2.0	2.00000000	2.00000000	2.00000000	2.00000000	2.00000000

11.3.2 非线性问题的差分方法

对于一般的二阶方程非线性边值问题(11.1.1)，(11.1.2)，可类似地建立差分方程．为保证边值问题存在唯一解，设 f 应满足定理(11.1.4)的条件，并设

$$L=\max_{(x,y,y')\in D}\left|\frac{\partial f}{\partial y'}(x,y,y')\right|$$

其中 D 域如(11.1.4)作的规定．

将(11.3.5)，(11.3.6) 用于方程(11.1.1)，并略去误差项，加上边界条件，可得到边值问题解在节点近似值 y_0，y_1，…，y_N 满足的非线性方程组：

$$\begin{cases} y_0=\alpha \\ \dfrac{y_{i+1}-2y_i+y_{i-1}}{h^2}-f\left(x_i,y_i,\dfrac{y_{i+1}-y_{i-1}}{2h}\right)=0 \\ \qquad\qquad (i=1,2,\cdots,N-1) \\ y_N=\beta \end{cases} \tag{11.3.14}$$

11.3.15 定理 若边值问题(11.1.1)，(11.1.2) 满足定理(11.1.4)的条件，并且 $L=\max\limits_{(x,y,y')\in D}\left|\dfrac{\partial f}{\partial y'}(x,y,y')\right|$，如果 $h\leqslant\dfrac{2}{L}$，则(11.3.14)有唯一的解．

解(11.3.14)的数值方法，可以选用解非线性方程组的 Newton 方法（见第 9 章）：

将(11.3.14)中 $y_0=\alpha$，$y_N=\beta$代入$i=1$和$i=N-1$的方程中，未知数写成向量 $\boldsymbol{y}=(y_1,\ y_2,\ \cdots,\ y_{N-1})^T$，则 Newton 法迭代公式为

$$\boldsymbol{y}^{(k)}=\boldsymbol{y}^{(k-1)}-J(\boldsymbol{y}^{(k-1)})^{-1}\boldsymbol{F}(\boldsymbol{y}^{(k-1)})$$

其中 J ($\boldsymbol{y}$) 是方程组 Jacobi 矩阵，写成

$$
J(y)=\begin{pmatrix}
2+h^2 f_y\left(x_1,y_1,\frac{y_2-\alpha}{2h}\right) & -1+\frac{h}{2}f_{y'}\left(x_1,y_1,\frac{y_2-\alpha}{2h}\right) & & 0\\
-1-\frac{h}{2}f_{y'}\left(x_2,y_2,\frac{y_3-y_1}{2h}\right) & 2+h^2 f_y\left(x_2,y_2,\frac{y_3-y_1}{2h}\right) & \ddots & \\
 & \ddots & \ddots & \ddots \\
0 & & -1-\frac{h}{2}f_{y'}\left(x_{N-1},y_{N-1},\frac{\beta-y_{N-2}}{2h}\right) & 2+h^2 f_y\left(x_{N-1},y_{N-1},\frac{\beta-y_{N-2}}{2h}\right)
\end{pmatrix}
$$

所以每迭代一步都要解一个三对角方程组

$$
J(y)\begin{pmatrix} v_1\\ v_2\\ \vdots\\ v_{N-2}\\ v_{N-1}\end{pmatrix}=-\begin{pmatrix}
2y_1-y_2-\alpha+h^2 f\left(x_1,y_1,\frac{y_2-\alpha}{2h}\right)\\
-y_1+2y_2+y_3+h^2 f\left(x_2,y_2,\frac{y_3-y_1}{2h}\right)\\
\vdots\\
-y_{N-3}+2y_{N-2}-y_{N-1}+h^2 f\left(x_{N-2},y_{N-2},\frac{y_{N-1}-y_{N-3}}{2h}\right)\\
-y_{N-2}+2y_{N-1}-\beta+h^2 f\left(x_{N-1},y_{N-1},\frac{\beta-y_{N-2}}{2h}\right)
\end{pmatrix}
$$

其中系数矩阵和右端向量中的 y_i 均取 $y_i^{(k-1)}$。解出 v_i 后，令 $y_i^{(k)}=y_i^{(k-1)}+v_i$ $(i=1, 2, \cdots, N-1)$，即完成一次迭代。

11.4

11.4 变分方法

在实际问题中十分重要的一类二阶线性方程的边值问题是“自伴随”的边值问题。一般，它的方程为

11.4.1 $$-\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right)+q(x)y=f(x), a\leqslant x\leqslant b$$

其中

$p\in C^1[a,b], p(x)\geqslant p_0>0$

$q\in C[a,b], q(x)\geqslant 0$

在 $x=a$ 和 $x=b$ 的边界条件，可以分别取为第一、二、三类边界条件，如(11.1.2)，(11.1.3)所示。

(11.4.1) 的物理背景可理解为弹性杆的平衡问题，其中 y 为杆的纵向位移。在第一类边界条件的情形，杆的总能量为

11.4.2 $$I(y)=\frac{1}{2}\int_a^b\left[p(x)\left(\frac{dy}{dx}\right)^2+q(x)(y(x))^2 - 2f(x)y(x)\right]dx$$

若给定函数 y，$I(y)$就确定一个实数值，故称 I 为一个“泛函”。

11.4.1 变分问题

对方程(11.4.1)的第一边值问题

11.4.3 边值问题

$$-\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right)+q(x)y=f(x), \quad a\leqslant x\leqslant b$$

$y(a)=y_a$, $y(b)=y_b$

可以提与之相关的泛函极小值问题，即

11.4.4 Ritz（里兹）变分问题（Ritz's Variational Problem） 求 $y\in K$，使得对一切 $w\in K$，有 $I(y)\leqslant I(w)$，其中

11.4.5 $K=\{y\mid y\in C^1[a,b],y(a)=y_a,y(b)=y_b\}$

$I(y)$ 如（11.4.2)所示. 也可以提

11.4.6 Галёркин（伽辽金）变分问题 / Галёркин Variational Problem 求 $y\in K$,使得 $D(y,w)-F(w)=0$ 对一切 $w\in K_0$ 成立，其中

11.4.7 $$D(y,w)=\int_a^b\left[p(x)\frac{dy}{dx}\frac{dw}{dx}dx+q(x)yw\right]dx$$

11.4.8 $$F(w)=\int_a^b f(x)w\ dx$$

K 如（11.4.5）所示，而

11.4.9 $K_0=C_0^1[a,b]$

$$=\{y\mid y\in C^1[a,b],y(a)=0,y(b)=0\}$$

如果边值问题（11.4.3）的物理背景理解为杆的平衡问题，则 Ritz变分问题可理解为**最小势能原理**，Галёркин 变分问题则可理解为**虚功原理**. 三者有如下关系.

11.4.10 定理 如 $y(x)$ 满足边值问题(11.4.3)，则 y 满足 Галёркин变分问题(11.4.6)，反之，若 y 满足 Галёркин变分问题（11.4.6)，且 $y\in C^1[a,b]\cap C^2(a,b)$,则 y 满足边值问题(11.4.3).

如果 y 满足 Ritz变分问题(11.4.4)，则 y 满足Галёркин 变分问题(11.4.6)，反之亦然.

为了简化问题，设(11.4.3)中的 $y_a=y_b=0$，这样在变分问题中，$K=K_0$. 如果 $y(x)$ 满足非齐次的边界条件,即

$y_a \neq 0$, $y_b \neq 0$，则令

$$\tilde{y}(x) = y(x) - y_a - \frac{y_b - y_a}{b-a}(x-a)$$

可验证$\tilde{y}(a) = \tilde{y}(b) = 0$. $\tilde{y}(x)$ 满足齐次边界条件的边值问题.

定理（11.4.10）指出的Ritz变分问题(11.4.4)与Галёркин变分问题（11.4.6）的等价性，仅对“自伴随”边值问题成立，即方程(11.4.1)中 $p(x) \geqslant p_0 > 0$，$q \geqslant 0$，此时有 $D(y, w) = D(w, y)$，且对任意的 $y \in C^1[a, b]$，$D(y, y) \geqslant 0$. 对于一些非自伴随的边值问题，仍可有Галёркин变分问题，所以 Галёркин 变分问题使用更为广泛.

由于在求解边值问题(11.4.3)时（例如用差分方法），要处理 y 的二阶导数，而求解变分问题（11.4.4）或（11.4.6）时，只要处理 y 的一阶导数. 有时把(11.4.6）称为(11.4.3）的弱形式.

对于第二或第三类边值问题，也可提出类似的变分问题.

11.4.11 例 若边值问题为（11.4.1），边界条件

$$y(a) = 0\text{，}\frac{dy(b)}{dx} = 0$$

则仍有类似的 Ritz 变分问题和 Галёркин 变分问题，其中 $I(y)$仍为(11.4.2)所示，而 $D(y, w)$，$F(w)$仍如(11.4.7)，(11.4.8)所示，但

$$K = K_0 = \{y \mid y \in C^1[a,b], y(a) = 0\}$$

在例（11.4.11)的变分问题中，并没有要求函数集合K中的函数满足第二类边界条件$\frac{dy(b)}{dx} = 0$，因而对变分问题的解 y 也没有提此要求. 但是，类似定理(11.4.10)，如果 y 满足变分问题，且 $y \in C^1[a, b] \cap C^2(a, b)$，则 y 满足

边值问题，所以变分问题的解是自然满足第二类边界条件的。这时条件$\frac{dy(b)}{dx}=0$称**自然边界条件**（Natural Boundary Condition）。而第一类边界条件$y(a)=0$称为**强加边界条件**（Forced Boundary Condition），或**约束边界条件**。

11.4.12 例 若边值问题为方程（11.4.1）及边界条件

$$\left[-p(x)\frac{dy}{dx}+\alpha_1 y(x)\right]_{x=a}=g_1$$

$$\left[p(x)\frac{dy}{dx}+\alpha_2 y(x)\right]_{x=b}=g_2 \qquad \alpha_1\geqslant 0,\alpha_2\geqslant 0$$

则在Ritz变分问题和Галёркин 变分问题中

$$D(y,w)=\int_a^b\left[p(x)\frac{dy}{dx}\frac{dw}{dx}dx+q(x)yw\right]dx +\alpha_1 y(a)w(a)+\alpha_2 y(b)w(b)$$

$$F(w)=\int_a^b f(x)wdx+g_1w(a)+g_2w(b)$$

$$I(y)=\frac{1}{2}D(y,y)-F(y)$$

本例中$K=K_0=C^1[a,b]$，两端边界条件都是自然边界条件。

至于混合边界条件的边值问题，例如，一端给出第一类条件，另一端给出第三类条件，则可类似写出对应的变分问题。

11.4.2 变分问题的近似计算

用变分方法解边值问题（11.4.3），就是根据定理（11.4.10）的等价关系，求解 Ritz 变分问题或 Галеркин 变分问题。

以上的 Ritz 变分问题可以归纳为

11.4.13 变分问题 求$y \in K$，使得对一切$w \in K$，$I(y) \leqslant I(w)$成立。其中

$$I(w)=\frac{1}{2}D(w,w)-F(w)$$

K为满足一定条件的函数集合，$D(y, w)$为（11.4.7）及例（11.4.12）中的积分式，满足$D(y,w)=D(w, y)$。

Ritz 方法的近似计算，就是找到函数集合K的一个子集合S_N。S_N的一组基是$\{\varphi_1, \varphi_2, \cdots, \varphi_N\}$，即$S_N$中任一元素$w_N$，均可表示成

$$w_N=c_1\varphi_1+c_2\varphi_2+\cdots+c_N\varphi_N$$

以S_N代替K，可得到近似变分问题，

11.4.14 近似变分问题 求$y_N \in S_N$，使对一切$w_N \in S_N$，$I(y_N) \leqslant I(w_N)$成立。

（11.4.14）的求解，只要先把$I(w_N)$列出，有：

$$I(w_N)=\frac{1}{2}D\left(\sum_{i=1}^{N}c_i\varphi_i, \sum_{j=1}^{N}c_j\varphi_j\right)-F\left(\sum_{i=1}^{N}c_i\varphi_i\right)$$

$$=\frac{1}{2}\sum_{i,j=1}^{N}D(\varphi_i,\varphi_j)c_ic_j-\sum_{i=1}^{N}c_iF(\varphi_i)$$

然后对$I(w_N)$求极小，即$\frac{\partial I(w_N)}{\partial c_j}=0$，$j=1, \cdots, N$，得到$c_1, \cdots, c_N$的一个方程组。所以，Ritz 方法的求解过程如下：

11.4.15 Ritz 方法

1° 选取K的子集合S_N，它的一组基函数是$\{\varphi_1, \varphi_2, \cdots, \varphi_N\}$。

2° 求解方程组

11.4.16 $$\sum_{j=1}^{N}D(\varphi_i,\varphi_j)c_j-F(\varphi_i)=0 \qquad (i=1,2,\cdots,N)$$

得到解$\widetilde{c}_1$，$\widetilde{c}_2$，…，$\widetilde{c}_N$.

3° 令$y_N=\widetilde{c}_1\varphi_1+\widetilde{c}_2\varphi_2+\cdots+\widetilde{c}_N\varphi_N$，作为变分问题(11.4.13)的近似解.

上面的Галёркин变分问题可以归纳为

11.4.17 变分问题 求$y\in K$，使得$D(y, w)-F(w)=0$对一切$w\in K$成立.

与Ritz方法类同，找K的子集合S_N，S_N中任一元素$w_N=\sum_{j=1}^{N}c_j\varphi_j$. 则得

11.4.18 近似变分问题 求$y_N\in S_N$，使得

$D(y_N, w_N)-F(w_N)=0$对一切$w_N\in S_N$成立.

将$y_N=\sum_{j=1}^{N}\widetilde{c}_j\varphi_j$，$w_N=\sum_{i=1}^{N}c_i\varphi_i$代入(11.4.18)，

得到

$$D(y_N,w_N)-F(w_N)$$

$$=\sum_{i=1}^{N}\left[\sum_{j=1}^{N}D(\varphi_i,\varphi_j)\widetilde{c}_j-F(\varphi_i)\right]c_i=0$$

对任意的$(c_1, c_2, \cdots, c_N)$成立，即

$$\sum_{j=1}^{N}D(\varphi_i,\varphi_j)\widetilde{c}_j-F(\varphi_i)=0\quad(i=1,2,\cdots,N)$$

它与方程组(11.4.16)是相同的，所以，对于自伴随的边值问题Галёркин方法的求解过程，和Ritz方法求解过程是一致的.

11.4.19 例 用变分方法近似求解边值问题

$$-\frac{d^2y}{dx^2}=x^2,\quad 0\leqslant x\leqslant 1$$

$$y(0)=0,\quad y(1)=0$$

解 本题变分问题中

$K=\{y \mid y\in C^1[0,1], y(0)=0, y(1)=0\}$

$$D(y,w)=\int_0^1 \frac{dy}{dx}\frac{dw}{dx}dx,\quad F(w)=\int_0^1 x^2 w\,dx$$

取K中子集合S_N为多项式的集合，为了满足边界条件，S_N中的函数取为

$w_N=x(1-x)(c_1+c_2x+\cdots+c_Nx^{N-1})$

对$N=1$，$\varphi_1(x)=x(1-x)$，方程组(11.4.16)只有一个方程

$D(\varphi_1,\varphi_1)c_1-F(\varphi_1)=0$

而 $D(\varphi_1,\varphi_1)=\int_0^1\left(\frac{d\varphi_1}{dx}\right)^2dx=\frac{1}{3}$

$F(\varphi_1)=\int_0^1 x^3(1-x)dx=\frac{1}{20}$

解出$\tilde{c}_1=\frac{3}{20}$，故近似解$y_1=\frac{3}{20}x(1-x)$。

对$N=2$，$\varphi_1(x)=x(1-x)$，$\varphi_2(x)=x^2(1-x)$，而$D(\varphi_1,\varphi_1)=\frac{1}{3}$，$D(\varphi_1,\varphi_2)=\frac{1}{6}$，$D(\varphi_2,\varphi_2)=\frac{2}{15}$，$F(\varphi_1)=\frac{1}{20}$，$F(\varphi_2)=\frac{1}{30}$。因此，方程组(11.4.16)为

$$\begin{pmatrix}\frac{1}{3} & \frac{1}{6}\\ \frac{1}{6} & \frac{2}{15}\end{pmatrix}\begin{pmatrix}c_1\\ c_2\end{pmatrix}=\begin{pmatrix}\frac{1}{20}\\ \frac{1}{30}\end{pmatrix}$$

解出$\tilde{c}_1=\frac{1}{15}$，$\tilde{c}_2=\frac{1}{6}$，故近似解$y_2=\frac{1}{30}x(1-x)(2+5x)$。

本例精确解为 $y=\frac{x}{12}(1-x^3)$，下表列出几个点上y，

y_1 及 y_2 之值.

x	0	0.25	0.5	0.75	1
y_1	0	0.0281	0.0375	0.0281	0
y_2	0	0.0203	0.0375	0.0359	0
y	0	0.0205	0.0365	0.0361	0

在例(11.4.19)的变分近似计算中，S_N 取的是多项式函数的集合. 还有别的取法，例如取成在$[a, b]$的某种剖分下与节点有关的分段多项式的集合，这就是有限元方法（见11.5）.

11.5

11.5 有限元方法

有限元方法 (Finite Element Method) 可以看成是一种变分方法. 考虑边值问题

11.5.1 $$-\frac{d}{dx}\left(p(x)\frac{dy}{dx}\right)+q(x)y=f,\quad a\leqslant x\leqslant b$$

11.5.2 $$y(a)=0,\qquad p(b)\frac{dy(b)}{dx}+\alpha y(b)=g$$

其中$p\in C^1[a,b]$，$q,f\in C[a,b]$，$p(x)\geqslant p_0>0$，$q(x)\geqslant 0$，$\alpha\geqslant 0$. 对应的 Галёркин 变分问题是

11.5.3 求 $y\in K$，使得 $D(y, w)-F(w)=0$ 对一切$w\in K$成立，其中

$$K=\{y\mid y\in C^1[a,b],\ y(a)=0\}$$

$$D(y,w)=\int_a^b\left(p\frac{dy}{dx}\frac{dw}{dx}+qyw\right)dx+\alpha y(b)w(b)$$

$$F(w)=\int_a^b fw dx + gw(b)$$

11.5.1 线性元

在变分问题使用Ritz方法求近似解的过程(11.4.15)中，如果选 S_N 为分段线性函数，则构成最简单的有限元方法，即使用线性元的有限元方法。

在$[a, b]$中插入节点：$a=x_0<x_1<\cdots<x_N=b$。每个子区间 $e_i=[x_{i-1}, x_i]$ $(i=1, 2\cdots, N)$ 称为"单元"，它的长度记为$h_i=x_i-x_{i-1}$。对应于每个节点x_i，有分段线性插值基函数 $\varphi_i(x)$：

11.5.4
$$\varphi_0(x)=\begin{cases}\dfrac{x_1-x}{h_1} & 0\leqslant x\leqslant x_1\\ 0 & x_1\leqslant x\end{cases}$$

$$\varphi_i(x)=\begin{cases}0 & x\leqslant x_{i-1}\\ \dfrac{x-x_{i-1}}{h_i} & x_{i-1}\leqslant x\leqslant x_i\\ \dfrac{x_{i+1}-x}{h_{i+1}} & x_i\leqslant x\leqslant x_{i+1}\\ 0 & x_{i+1}\leqslant x\end{cases}\qquad (i=1,2,\cdots,N-1)$$

$$\varphi_N(x)=\begin{cases}0 & x\leqslant x_{N-1}\\ \dfrac{x-x_{N-1}}{h_N} & x_{N-1}\leqslant x\leqslant x_N\end{cases}$$

设 S_N 中的函数为这些基函数的线性组合，若 $w\in S_N$，有

11.5.5 $w(x)=w_0\varphi_0(x)+w_1\varphi_1(x)+\cdots+w_N\varphi_N(x)$

以(11.5.5)形式的函数去近似边值问题(11.5.1)，(11.5.2)的解，由于基函数(11.5.4)的性质，不难验证(11.5.5)

中的系数 $w_i=w(x_i)$ $(i=0,\ 1,\ \cdots N)$．如果让 $w(x)$满足边界条件(11.5.2)，则有 $w_0=0$。

选好基函数之后，可写出近似变分问题（用Галёркин形式写出）：

11.5.6　求 $y\in S_N$，使得

$$\sum_{i=1}^{N}\int_{e_i}\left(p\frac{dy}{dx}\frac{dw}{dx}+qyw\right)dx+\alpha y(b)w(b)$$

$$=\sum_{i=1}^{N}\int_{e_i}fwdx+gw(b)$$

对一切 $w\in S_N$ 成立。可由此列出方程(11.4.16)，但一般有限元方法计算过程如下。

11.5.7　令

$$\{w\}_{e_i}=\begin{bmatrix}w_{i-1}\\ w_i\end{bmatrix},\quad \{y\}_{e_i}=\begin{bmatrix}y_{i-1}\\ y_i\end{bmatrix}$$

$$[N]=\left[\frac{x_i-x}{h_i},\ \frac{x-x_{i-1}}{h_i}\right]=[N_i(x),M_i(x)]$$

$$[B]=\left[-\frac{1}{h_i},\ \frac{1}{h_i}\right]$$

则在(11.5.6)中

$$\int_{e_i}\left(p\frac{dy}{dx}\frac{dw}{dx}+gyw\right)dx=\{w\}_{e_i}^T[K]_{e_i}\{y\}_{e_i}$$

其中

11.5.8　$$[K]_{e_i}=\int_{x_{i-1}}^{x_i}(p[B]^T[B]+q[N]^T[N])dx$$

$$=\begin{bmatrix}k_{i-1,\,i-1}^{e_i} & k_{i-1,\,i}^{e_i}\\ k_{i,\,i-1}^{e_i} & k_{ii}^{e_i}\end{bmatrix}$$

11.5.9 $k^{e_i}_{i-1,\,i-1} = \frac{1}{h_i^2}\int_{x_{i-1}}^{x_i} p dx + \int_{x_{i-1}}^{x_i} qN_i^2 dx$

$$k^{e_i}_{i-1,\,i} = k^{e_i}_{i,\,i-1} = -\frac{1}{h_i^2}\int_{x_{i-1}}^{x_i} p dx + \int_{x_{i-1}}^{x_i} qN_iM_i dx$$

$$k^{e_i}_{i,\,i} = \frac{1}{h_i^2}\int_{x_{i-1}}^{x_i} p dx + \int_{x_{i-1}}^{x_i} qM_i^2 dx$$

$$(i=1,2,\cdots,N-1)$$

为了将(11.5.6)左端最后一项写进矩阵式，当 $i=N$时，$[K]e_N$ 的元素$k^{e_N}_{NN}$应为

11.5.10 $k^{e_N}_{NN} = \frac{1}{h_N^2}\int_{x_{N-1}}^{x_{N-1}} p dx + \int_{x_{N-1}}^{x_N} qM_N^2 dx + a$

而$[K]e_N$ 其它三个元素同(11.5.9).

类似地处理(11.5.6)的右端项，有

$$\int_{e_i} fw dx = \{w\}^T_{e_i}\{F\}_{e_i}$$

其中

11.5.11 $\{F\}_{e_i} = \begin{bmatrix} F^{e_i}_{i-1} \\ F^{e_i}_i \end{bmatrix}$

11.5.12 $F^{e_i}_{i-1} = \int_{x_{i-1}}^{x_i} N_i f dx$

$$F^{e_i}_i = \int_{x_{i-1}}^{x_i} M_i f dx \quad (i=1,2,\cdots,N-1)$$

11.5.13 $F^{e_N}_N = \int_{x_{i-1}}^{x_i} M_N f dx + g$, $F^{e_N}_{N-1}$ 同(11.5.12)

11.5.14 **定义/单元刚度矩阵/Element Stiffness Matrix**

(11.5.8)—(11.5.10) 确定的 $[K]_{e_i}$ 称为用线性元求解 (11.5.1), (11.5.2)的**单元刚度矩阵**.

11.5.15 **定义/单元荷载向量/Element Load Vector** (11.5.11)—(11.5.13)确定的 $\{F\}_{e_i}$ 称为用线性元求解 (11.5.1), (11.5.2)的**单元荷载向量**.

令 $\{y\}=(y_0,y_1,\cdots,y_N)^T$, $\{w\}=(w_0,w_1,\cdots,w_N)^T$

$$
[\overline{K}]_{e_i}=\begin{bmatrix} & \vdots & \vdots & \\ \cdots & k^{e_i}_{i-1,i-1} & k^{e_i}_{i-1,i} & \cdots \\ \cdots & k^{e_i}_{i,i-1} & k^{e_i}_{i,i} & \cdots \\ & \vdots & \vdots & \end{bmatrix}\begin{matrix} \\ 第i-1行 \\ 第i行 \\ \\ \end{matrix}
$$

第 $i-1$ 列　　第 i 列

$$\{\overline{F}\}_{e_i}=[\cdots,F^{e_i}_{i-1},\ F^{e_i}_{i},\cdots]^T$$

在 $[\overline{K}]_{e_i}$ 和 $\{\overline{F}\}_{e_i}$ 中, … 代表零元素.

11.5.16 **定义/刚度矩阵/Stiffness Matrix** $[K]=\sum_{i=1}^{N}[\overline{K}]_{e_i}$ 称为解(11.5.1), (11.5.2)的刚度矩阵, 或称**总刚度矩阵**.

11.5.17 **定义/荷载向量/Load Vector** $\{F\}=\sum_{i=1}^{N}\{\overline{F}\}_{e_i}$ 称为解(11.5.1), (11.5.2)的荷载向量, 或称**总荷载向量**.

$[K]$ 是一个 $(N+1)\times(N+1)$ 的矩阵, 它由 $[\overline{K}]_{e_i}$ 迭加而成, 而 $[\overline{K}]_{e_i}$ 是单元刚度矩阵 $[K]_{e_i}$ 的一种"放大", 即将 $[K]_{e_i}$ 的四个元素放到 $(N+1)\times(N+1)$ 矩阵 $[\overline{K}]_{e_i}$ 相应的位置, 而令 $[\overline{K}]_{e_i}$ 其它元素为0. 同理, 对 $\{F\}$ 的构成可作类似的分析.

考虑到边界条件(11.5.2)中 $y(a)=0$，所以在向量$\{y\}$和$\{w\}$中，应取 $y_0=w_0=0$. 这样近似变分问题可写成矩阵形式

$$(0,w_1,\cdots,w_N)^T\left([K]\begin{pmatrix}0\\y_1\\\vdots\\y_N\end{pmatrix}-\{F\}\right)=0$$

对一切$(w_1,\ \cdots,\ w_N)$成立.或改写为

$$11.5.18\quad (w_1,\cdots,w_N)^T\left([\widetilde{K}]\begin{pmatrix}y_1\\\vdots\\y_N\end{pmatrix}-\{\widetilde{F}\}\right)=0$$

对一切 $(w_1,\ \cdots,\ w_N)$ 成立. 其中 $[\widetilde{K}]$ 为$[K]$划去第一行和第一列而得到的 $N\times N$ 矩阵，$\{\widetilde{F}\}$为$\{F\}$划去第一个元素所得的向量. 由 $(w_1,\ \cdots w_N)$ 的任意性，可得到方程组

$$11.5.19\quad [\widetilde{K}]\begin{pmatrix}y_1\\\vdots\\y_N\end{pmatrix}-\{\widetilde{F}\}=0$$

11.5.20 **定理** 对应于边值问题(11.5.1)，(11.5.2)，方程组(11.5.19)的系数矩阵$[\widetilde{K}]$是一个正定矩阵，因此(11.5.19)存在唯一的解 $(y_1,\ \cdots,\ y_N)^T$，而$\sum_{j=1}^{N} y_j\varphi_j(x)$就是近似变分问题(11.5.6)的解，$y_i$是它在节点$x_i$之值$(i=1,2,\cdots,N)$.

可以求解一个与(11.5.19)等价的 $(N+1)$ 阶方程组

$$\begin{pmatrix} 1 & 0 & \cdots & 0 \\ 0 & & & \\ \vdots & & \widetilde{K} & \\ 0 & & & \end{pmatrix}\begin{pmatrix} y_0 \\ y_1 \\ \vdots \\ y_N \end{pmatrix}-\begin{pmatrix} 0 \\ \widetilde{F} \\ \\ \end{pmatrix}=0 \tag{11.5.21}$$

它的系数矩阵是将$[K]$的第一行和第一列元素作了改变，同时$\{F\}$的第一个元素也变为0。这样，求解(11.5.21)时，系数矩阵和$[K]$一样是$N+1$阶的，而求解(11.5.19)时，从$N+1$阶的$[K]$到N阶的$[\widetilde{K}]$要重新排列矩阵的元素。

11.5.22 **例** 用线性元求解边值问题

$$-\frac{d^2y}{dx^2}=2,\quad 0\leqslant x\leqslant 1$$

$$y(0)=0,\quad y'(1)=0$$

解 本例中$D(y,w)=\int_0^1 \frac{dy}{dx}\frac{dw}{dx}dx$，$F(w)=2\int_0^1 w dx$

将$[0,1]$等分为四个单元，节点为0，$\frac{1}{4}$，$\frac{1}{2}$，$\frac{3}{4}$，1。

取$h_i=h=\frac{1}{4}$。可得

$$[K]_{e_i}=\frac{1}{h}\begin{pmatrix} 1 & -1 \\ -1 & 1 \end{pmatrix}$$

$$\{F\}_{e_i}=h\begin{pmatrix} 1 \\ 1 \end{pmatrix}\quad (i=1,2,3,4)$$

$$[K]=\sum_{i=1}^{4}[\widetilde{K}]_{e_i}=\frac{1}{h}\begin{pmatrix} 1 & -1 & & & \\ -1 & 1+1 & -1 & & \\ & -1 & 1+1 & -1 & \\ & & -1 & 1+1 & -1 \\ & & & -1 & 1 \end{pmatrix}$$

$$\{F\} = \sum_{i=1}^{4}\{\overline{F}\}_{e_i} = h\begin{pmatrix} 1 \\ 1+1 \\ 1+1 \\ 1+1 \\ 1 \end{pmatrix}$$

经过约束处理，得到方程组

$$4\begin{pmatrix} 2 & -1 & 0 & 0 \\ -1 & 2 & -1 & 0 \\ 0 & -1 & 2 & -1 \\ 0 & 0 & -1 & 1 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix} = \frac{1}{4}\begin{pmatrix} 2 \\ 2 \\ 2 \\ 1 \end{pmatrix}$$

求解后得$y_1=\frac{7}{16}$，$y_2=\frac{12}{16}$，$y_3=\frac{15}{16}$，$y_4=1$。

11.5.2 高次元

线性元方法是将边值问题近似解用分段线性插值函数表示，而二次元方法则用分段二次插值函数表示。仿照一次元的方法，将插值基函数在每个元e_i上作出，其它的运算则类似线性元方法。

为了计算方便，将区间$x_{i-1}\leqslant x\leqslant x_i$变换到$-1\leqslant\xi\leqslant 1$。在$[-1,1]$上作二次插值，设插值点为$\xi=-1, 0, 1$，则二次插值基函数为

11.5.23 $N_1(\xi)=\frac{1}{2}\xi(\xi-1)$

$N_2(\xi)=1-\xi^2$

$N_3(\xi)=\frac{1}{2}\xi(\xi+1)$

设$[N]=[N_1(\xi), N_2(\xi), N_3(\xi)]$，$\{y\}_e=(y_1, y_2, y_3)^T$，则在$[-1, 1]$上，二次插值函数表示为

$y=[N]\{y\}$。

11.5.24 例 用二次元方法解

$$-\frac{d^2y}{dx^2}=1,\quad 0\leqslant x\leqslant 1$$

$$y(0)=1,\left(\frac{dy}{dx}+2y\right)_{x=1}=3$$

解 本例中 $D(y,w)=\int_0^1\frac{dy}{dx}\ \frac{dw}{dx}dx+2y(1)w(1)$

$$F(w)=\int_0^1 wdx+3w(1)$$

把〔0,1〕平分为两个单元，$e_1=\left[0,\frac{1}{2}\right]$，$e_2=\left[\frac{1}{2},\ 1\right]$，

$h_1=h_2=\frac{1}{2}$. 作变换

$$\xi=\frac{1}{h_i}[2x-(x_{i-1}+x_i)]\quad (i=1,2)$$

将 e_1，e_2 变换到标准单元 $-1\leqslant\xi\leqslant 1$，且

$$[B]=\frac{d}{dx}[N]=\frac{1}{h_i}[2\xi-1,-4\xi,2\xi+1]$$

可计算单元刚度矩阵

$$[K]_{e_1}=\int_{e_1}[B]^T[B]dx$$

$$=\int_{-1}^1[B]^T[B]\frac{h_1}{2}d\xi=\frac{1}{3}\begin{bmatrix}14 & -16 & 2\\ -16 & 32 & -16\\ 2 & -16 & 14\end{bmatrix}$$

由于 $D(y,w)$ 中存在边界项 $2y(1)w(1)$，所以在计算 $[K]_{e_2}$ 时，除和 $[K]_{e_1}$ 相同外，对角线的第三个元素加上2，得到

$$[K]_{e_2}=\frac{1}{3}\begin{pmatrix} 14 & -16 & 2 \\ -16 & 32 & -16 \\ 2 & -16 & 20 \end{pmatrix}$$

单元荷载向量为

$$\{F\}_{e_1}=\int_{-1}^{1} f[N]^T\cdot\frac{h_1}{2}dx=\frac{1}{3}\left(\frac{1}{4},1,\frac{1}{4}\right)^T$$

$$\{F\}_{e_2}=\frac{1}{3}\left(\frac{1}{4},1,\frac{37}{4}\right)^T$$

将单元的刚度矩阵及荷载向量分别对应迭加，得到

$$[K]=\frac{1}{3}\begin{pmatrix} 14 & -16 & 2 & 0 & 0 \\ -16 & 32 & -16 & 0 & 0 \\ 2 & -16 & 28 & -16 & 2 \\ 0 & 0 & -16 & 32 & -16 \\ 0 & 0 & 2 & -16 & 20 \end{pmatrix},\ [F]=\frac{1}{3}\begin{pmatrix} \frac{1}{4} \\ 1 \\ \frac{1}{2} \\ 1 \\ \frac{37}{4} \end{pmatrix}.$$

考虑到边界条件 $y_0=1$，在约束处理后得到方程组

$$\frac{1}{3}\begin{pmatrix} 32 & -16 & 0 & 0 \\ -16 & 28 & -16 & 2 \\ 0 & -16 & 32 & -16 \\ 0 & 2 & -16 & 20 \end{pmatrix}\begin{pmatrix} y_1 \\ y_2 \\ y_3 \\ y_4 \end{pmatrix}=\frac{1}{3}\begin{pmatrix} 17 \\ -\frac{3}{2} \\ 1 \\ \frac{37}{4} \end{pmatrix}$$

求解后得 $y_0=1$, $y_1=\frac{39}{32}$, $y_2=\frac{11}{8}$, $y_3=\frac{47}{32}$, $y_4=\frac{3}{2}$.

将单元 e_i 变换到 $-1\leqslant\xi\leqslant1$，并记 $y_i=y|_{x=x_{i-1}}$,

$$y_1{}'=\left.\frac{dy}{dx}\right|_{x=x_{i-1}},\quad y_2=y\,|_{x=x_i},\quad y_2{}'=\left.\frac{dy}{dx}\right|_{x=x_i}$$

令

11.5.25 $N_1(\xi)=\frac{1}{4}(\xi^3-3\xi+2), N_2(\xi)=\frac{1}{4}(-\xi^3+3\xi+2)$

$$M_1(\xi)=\frac{h_i}{2}\cdot\frac{1}{4}(\xi^3-\xi^2-\xi+1)$$

$$M_2(\xi)=\frac{h_i}{2}\cdot\frac{1}{4}(\xi^3+\xi^2-\xi-1.$$

并记

$[N]=[N_1(\xi)\ \ M_1(\xi)\ \ N_2(\xi)\ \ M_2(\xi)]$

$\{y\}_{e_i}=(y_1,y_1{}',y_2,y_2{}')^T$

则得到

11.5.26 单元 e_i 上三次 Hermite 插值函数

$y=[N]\{y\}_{e_i}$

用三次 Hermite 插值作有限元计算，每个节点 x_i 上都有两个未知数 y_i，$y_i{}'$。可以类似线性元方法列出其方程组解出 y_i，$y_i{}'$ $(i=0,\cdots,N)$ 后，近似解在每个单元如(11.5.25)所示。在全区间$[a,b]$，近似解有一阶导数的连续性。

11.6

11.6 样条函数方法

讨论线性边值问题

11.6.1 $y''=p(x)y'+q(x)y+r(x),\ a\leqslant x\leqslant b$

11.6.2 $\alpha_0 y'(a)-\alpha_1 y(a)=\alpha,\quad \beta_0 y'(b)+\beta_1 y(b)=\beta$

其中 $\alpha_0\alpha_1\geqslant 0, \beta_0\beta_1\geqslant 0$。

对$[a,b]$作剖分

$a=x_0<x_1<\cdots<x_n=b$

则$[a,b]$上以$\{x_j\}$为节点的三次样条函数可表示为：

11.6.3 $$s(x)=e+f(x-x_0)+\frac{1}{2}g(x-x_0)^2+\frac{1}{6}\sum_{j=0}^{n-1}d_j(x-x_j)_+^3$$

其中e，f，g，d_0，…，d_{n-1}为$n+3$个待定系数，

$$t_+=\begin{cases} t & ,当t\geqslant 0 \\ 0 & ,当t<0 \end{cases}$$

用形如(11.6.3)的$s(x)$作为边值问题近似解，不难对$s(x)$求出一、二阶导数，用节点x_i之值代入，考虑在节点x_i上满足(11.6.1)，便有

11.6.4 $$g+\sum_{j=0}^{n-1}d_j(x_i-x_j)_+=p(x_i)[f+g(x_i-x_0)+\frac{1}{2}\sum_{j=0}^{n-1}d_j(x_i-x_j)_+^2]+q(x_i)[e+f(x_i-x_0)+\frac{1}{2}g(x_i-x_0)^2+\frac{1}{6}\sum_{j=0}^{n-1}d_j(x_i-x_j)_+^3]+r(x_i.$$

$$(i=0,1,\cdots,n)$$

同理，由边界条件得到

11.6.5 $$\alpha_0 f-\alpha_1 e=\alpha$$

$$\beta_0[f+g(x_n-x_0)+\frac{1}{2}\sum_{j=0}^{n-1}d_j(x_n-x_j)_+^2]+\beta_1[e+f(x_n-x_0)+\frac{1}{2}g(x_n-{}_0)^2+\frac{1}{6}\sum_{j=0}^{n-1}d_j(x_n-x_j)_+^3]=\beta$$

(11.6.4)及(11.6.5)共有 $n+3$ 个方程，解出 e，f，g，d_0，…，d_{n-1}，代入(11.6.3)就得到边值问题的近似解。

11.6.6　例　用样条函数方法解边值问题

$y''+y+1=0$，　$0\leqslant x\leqslant 1$

$y(0)=0$，$y(1)=0$

解　把[0,1]分成三等分，节点为 $0,\frac{1}{3},\frac{2}{3},1$。(11.6.3)中有六个未知数 e，f，g，d_0，d_1，d_2。由边界条件 $y(0)=0$，得到 $e=0$，所以设

11.6.7　$s(x)=fx+gx^2+d_0x_+^3+d_1(x-\frac{1}{3})_+^3+d_2(x-\frac{2}{3})_+^3$

求出 $s'(x)$ 及 $s''(x)$，代入方程，并取 x 为 0，$\frac{1}{3}$，$\frac{2}{3}$，1，及边界条件 $y(1)=0$，可得到

$2g=-1$

$55d_0+57g+9f=-27$

$55d_1+116d_0+66g+18f=-27$

$55d_2+116d_1+189d_0+81g+275f=-27$

$d_2+8d_1+27d_0+27g+27f=0$

求解后得　$f=0.540814$，$g=-0.5$，$d_0=-0.061224$

$d_1=0.061224$，$d_2=0.061224$

将其代入(11.6.7)式即得边值问题近似解。

12

第12章 偏微分方程数值解法

12.1

12.1 椭圆型方程的差分解法

12.1.1 典型问题

椭圆型方程（Elliptic Equations）中最简单的典型方程是Laplace（拉普拉斯）方程

12.1.1 $$Lu=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

和Poisson（泊松）方程

12.1.2 $$Lu=\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=f(x,y)$$

椭圆型方程的主要定解问题是边值问题（Boundary Value Problems），即要求定出未知函数u，使得它在某个区域Ω内满足微分方程（12.1.1）或（12.1.2），并在区域的边界$\partial\Omega$上满足给定的边界条件（Boundary Conditions）边界条件可以由下面三种方式给出：

12.1.3 $$u|_{\partial\Omega}=\alpha(x,y)$$

12.1.4 $$\left.\frac{\partial u}{\partial n}\right|_{\partial\Omega}=\beta(x,y)$$

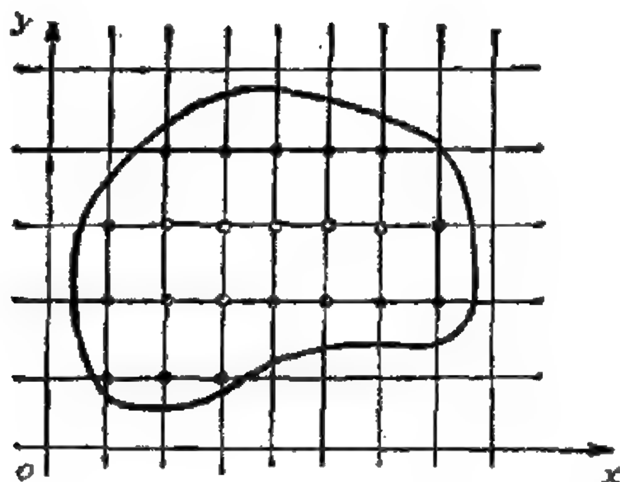
12.1.5 $$\left.\left(\frac{\partial u}{\partial n}+ku\right)\right|_{\partial\Omega}=\gamma(x,y)$$

其中n表示外法线方向，$k=k(x,y)\geqslant 0$，边界条件（12.1.3），（12.1.4）和（12.1.5）分别称为第一类，第二类和第三类边界条件。

12.1.2 网格和差分近似

设Ω为x，y平面上的一有界区域，$\partial\Omega$ 为其边界（见图12.1.6）。

12.1.6 图



取定沿x轴方向步长为h_1，y轴方向步长为h_2，作两族与坐标轴平行的直线：

$x=ih_1 \quad (i=0,\pm1,\pm2,\cdots)$

$y=jh_2 \quad (j=0,\pm1,\pm2,\cdots)$

两族直线的交点(ih_1,jh_2)称为**网格点**或节点，记为(x_i,y_j)。仅考虑属于$\Omega\cup\partial\Omega$的网格点。如果两个网格点(x_i,x_j)和$(x_{i'},y_{j'})$满足下面关系式

$$\left|\frac{x_i-x_{i'}}{h_1}\right|+\left|\frac{y_j-y_{j'}}{h_2}\right|=1$$

那么称这两个网格点是**相邻的**。

如果一个节点的四个相邻的节点都属于$\Omega\cup\partial\Omega$，则称此节点为**内部节点**，或简称为内点。全部内点的全体记作Ω_h。如果一个节点的四个相邻的节点至少有一个不属于$\Omega\cup\partial\Omega$，则称其为**边界节点**，或简称为界点。全体界点的集合记为$\partial\Omega_h$。在图(12.1.6)中，以“•”表示界点，以“∘”表示内点。利用Taylor级数展开

$$\frac{1}{2h_1}\left[u(x_{i+1},y_j)-u(x_{i-1},y_j)\right]$$

$$=\frac{\partial u(x_i,y_j)}{\partial x}+\frac{1}{6}h_1^2\frac{\partial^3 u(x_i,y_j)}{\partial x^3}+O(h_1^4)$$

有

12.1.7 $$\left(\frac{\partial u}{\partial x}\right)_{ij}=\frac{\partial u(x_i,y_j)}{\partial x}\approx\frac{u(x_{i+1},y_j)-u(x_{i-1},y_j)}{2h_1}$$

同样地有

12.1.8 $$\left(\frac{\partial u}{\partial y}\right)_{ij}=\frac{\partial u(x_i,y_j)}{\partial y}\approx\frac{u(x_i,y_{j+1})-u(x_i,y_{j-1})}{2h_2}$$

对于二阶偏导数，同样可用Taylor级数展开获得近似表达式，并且容易求得

12.1.9 $$\begin{cases}\left(\dfrac{\partial^2 u}{\partial x^2}\right)_{ij}=\dfrac{\partial^2 u(x_i,y_j)}{\partial x^2}\\ \qquad\approx\dfrac{u(x_{i+1},y_j)-2u(x_i,y_j)+u(x_{i-1},y_j)}{h_1^2}\\ \left(\dfrac{\partial^2 u}{\partial y^2}\right)_{ij}=\dfrac{\partial^2 u(x_i,y_j)}{\partial y^2}\\ \qquad\approx\dfrac{u(x_i,y_{j+1})-2u(x_i,y_j)+u(x_i,y_{j-1})}{h_2^2}\end{cases}$$

12.1.3 差分格式的构造方法

由Poisson方程(12.1.2)的**差分格式**(Finite Difference Scheme)构造方法，可直接得到 Laplace 方程（12.1.1）的差分格式。因此，仅讨论Poisson方程。取正方形网格：$h=h_1=h_2$，任取一内点（x_i,y_j），利用（12.1.9）可以由Poisson方程直接得到一近似表达式

$$\frac{u(x_{i+1},y_j)-2u(x_i,y_j)+u(x_{i-1},y_j)}{h^2}$$

$$+\frac{u(x_i,y_{j+1})-2u(x_i,y_j)+u(x_i,y_{j-1})}{h^2}\approx -f(x_i,y_j)$$

用u_{ij}表示$u(x_i,y_j)$的近似值，则由上式可以得到逼近 Poisson方程的差分格式

12.1.10 $$L_h u=\frac{u_{i+1,j}-2u_{ij}+u_{i-1,j}}{h^2}+\frac{u_{i,j+1}-2u_{ij}+u_{i,j-1}}{h^2}$$

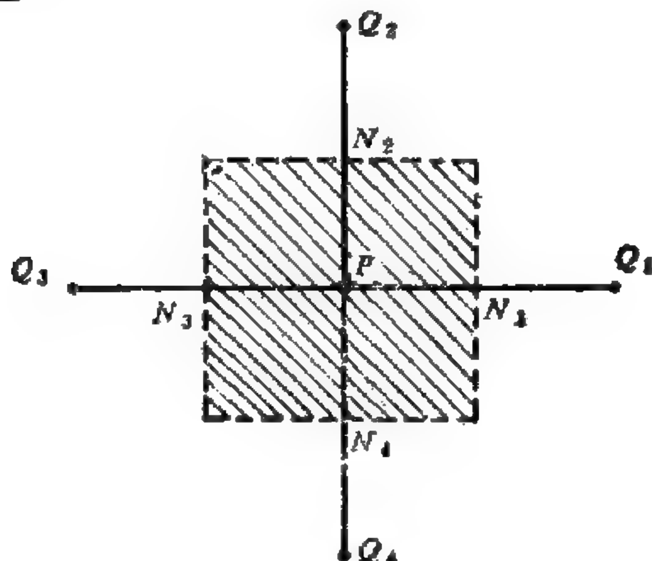
$$=-f_{ij}$$

其中$f_{ij}=f(x_i,y_j)$。

上述构造差分格式（12.1.10)的办法是直接用差商代替微商，这是最直观最方便的方法.差分格式(12.1.10)中只出现u在（x_i,y_j）及其四个邻点上的值，所以称其有**五点差分格式**（Five Point Difference Scheme）.

用**积分插值法**推导逼近Poisson方程的五点差分格式的步骤如下：设$P=(x_i,y_j)\in\Omega_h$，其四个邻点为$Q_1=(x_{i+1},y_j)$，$Q_2=(x_i,y_{j+1})$，$Q_3=(x_{i-1},y_j)$和$Q_4=(x_i,y_{j-1})$.令N_i为PQ_i的中点，$i=1,2,3,4$（见图12.1.11）.在有影区域D_{ij}上对Poisson方程（12.1.2）两边进行积分，有

12.1.11 图



$$\iint\limits_{D_{ij}} \frac{\partial^2 u}{\partial x^2}\, dxdy = \int_{y_{j-\frac{1}{2}}}^{y_{j+\frac{1}{2}}} \left(\frac{\partial u}{\partial x} \Bigg|_{x_{i-\frac{1}{2}}}^{x_{i+\frac{1}{2}}} \right) dy$$

其中$x_{i\pm\frac{1}{2}} = \frac{1}{2}(x_i + x_{i\pm 1})$，$y_{j\pm\frac{1}{2}} = \frac{1}{2}(y_j + y_{j\pm 1})$.

利用中点求积公式有

$$\iint\limits_{D_{ij}} \frac{\partial^2 u}{\partial x^2}\, dxdy \approx \left[\frac{\partial u(x_{i+\frac{1}{2}}, y_j)}{\partial x} - \frac{\partial u(x_{i-\frac{1}{2}}, y_j)}{\partial x} \right] h$$

把上式右边的微商用中心差商来代替，这样就得到

$$\iint\limits_{D_{ij}} \frac{\partial^2 u}{\partial x^2}\, dxdy \approx u_{i+1,j} - 2u_{ij} + u_{i-1,j}$$

同理

$$\iint\limits_{D_{ij}} \frac{\partial^2 u}{\partial y^2}\, dxdy \approx u_{i,j+1} - 2u_{ij} + u_{i,j-1}$$

此外，

$$\iint\limits_{D_{ij}} f(x,y)\, dxdy \approx f_{ij} h^2$$

综合上述各近似式，可以得到逼近Poisson 方程的差分格式

$$L_h u = \frac{1}{h^2}\left[(u_{i+1,j} - 2u_{ij} + u_{i-1,j}) + (u_{i,j+1} - 2u_{ij} + u_{i,j-1}) \right]$$

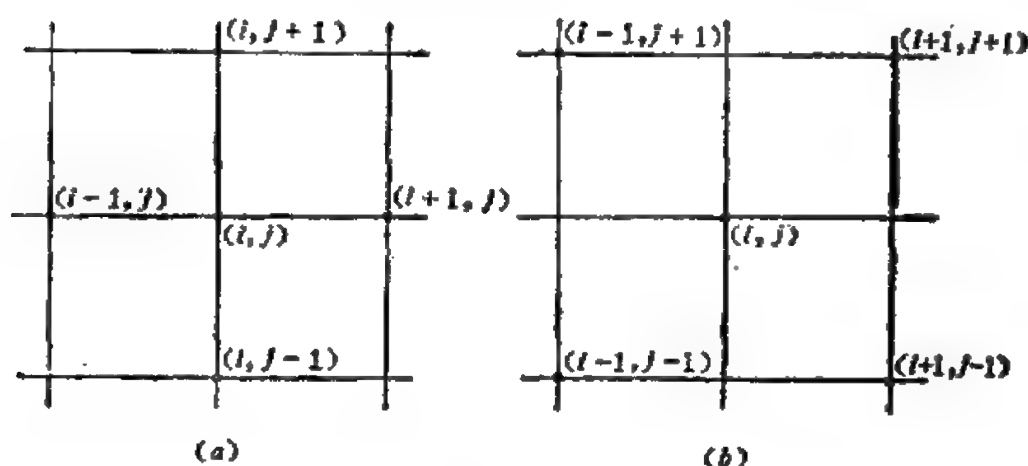
$$= -f_{ij}$$

这就是前面推导出来的五点差分格式，这个方法也称为**盒式积分法**. 利用这个方法构造变系数方程以及间断系数的方程的差分方法较为方便.

12.1.4 通用的差分格式

五点差分格式(12.1.10)是经常使用的，它的节点排列以 (x_i,y_j) 为中心，其余四个点是 (x_i,y_j) 的上、下、左、右

12.1.12 图



四个相邻的节点(见图12.1.12a)。如果不采用上述节点，而采用图（12.1.12b）的节点，也可以得到逼近 Poisson 方程的差分格式

12.1.13
$$L_h^{(1)}u=\frac{1}{2h^2}(u_{i+1,j+1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i-1,j-1}-4u_{ij})=-f_{ij}$$

利用差分格式（12.1.13）和（12.1.10）进行线性组合可以得到逼近Poisson方程的另一差分格式

12.1.14
$$\begin{aligned}L_h^{(2)}u=\frac{1}{6h^2}\{&4(u_{i+1,j}+u_{i-1,j}+u_{i,j+1}+u_{i,j-1}-4u_{ij})\\&+(u_{i+1,j+1}+u_{i+1,j-1}+u_{i-1,j+1}+u_{i-1,j-1}-4u_{ij})\}\\=-[f_{ij}+\frac{1}{12}(&f_{i,j+1}+f_{i,j-1}+f_{i+1,j}+f_{i-1,j}-4f_{ij})]\end{aligned}$$

由于差分格式(12.1.10)和(12.1.13)都仅采用五个节点，而差分格式(12.1.14)采用了九个节点，因此一般也称差分

格式(12.1.14)为九点差分格式.对于一个差分格式，其基本的要求是差分方程逼近微分方程，这种逼近的近似程度的好坏可以用**截断误差**（Truncation Error）来描述。

设$u=u(x,y)$是Poisson方程(12.1.2)的充分光滑的解，L_h是由(12.1.10)定义的**差分算子**(Difference Operator),称

12.1.15 $R_{ij}(u)=L_h u(x_i,y_j)-Lu(x_i,y_j)$

为差分格式(12.1.10)的截断误差。

实际计算截断误差时,只要将$L_h u(x_i,y_j)$的各项在(x_i,y_j)展成Taylor级数即可通过初等计算可以得到，五点差分格式（12.1.10）和(12.1.13）的截断误差是$O(h^2)$，而九点差分格式(12.1.14）的截断误差为$O(h^4)$。由于九点差分格式（12.1.14）截断误差的阶高，所以经常称其为**高精度格式**。显然，它更精确地逼近Poisson方程.在实际计算中，如果精度要求不是很高，以采用五点差分格式较为合适。

12.1.5 边界条件的处理

在对内点列出差分方程后，为了求解，还必须对边界条件进行处理以给出边界上的关系式。

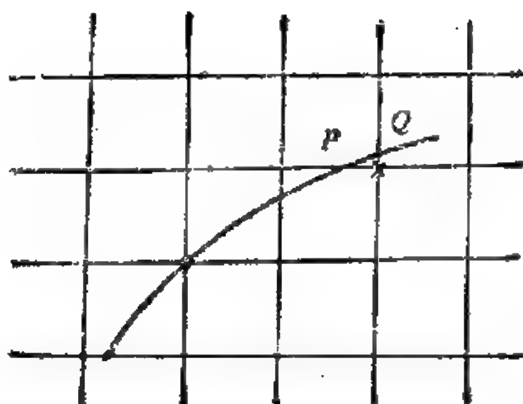
对第一类边界条件

$u|_{\partial\Omega}=\alpha(x,y)$ ，$(x,y)\in\partial\Omega$

在构造网格时，$\partial\Omega_h$通常都不与$\partial\Omega$相重合，可能有少数边界节点落在$\partial\Omega$上，而大部分边界节点不落在$\partial\Omega$上，由此而产生边界条件的转换：

1° 直接转移法.若边界节点(x_i,y_j)正好落在$\partial\Omega$上，如图(12.1.16)所示，则$u_{ij}=\alpha(x_i,y_j)$.如果(x_i,y_j)不在$\partial\Omega$上，此时$\partial\Omega$与网格线交于P和Q两点，则取与点(x_i,y_j)距离最近的点$\alpha(x,y)$的值为u_{ij}。

12.1.16 图

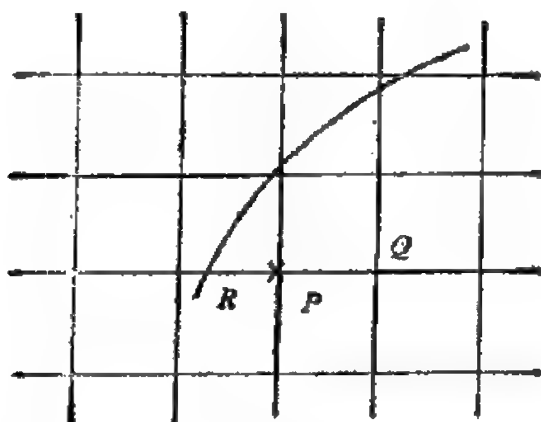


2° 线性插值．如图(12.1.17)所示，对于网格点P可用R,Q两点作线性插值

$$u(P)=\frac{h}{h+\delta}u(R)+\frac{\delta}{h+\delta}u(Q)$$

其中$\delta=RP<h$。

12.1.17 图



由于第二类边界条件可以看作第三类边界条件的特殊情况，所以仅处理第三类边界条件

$$\left(\frac{\partial u}{\partial n}+ku\right)\Big|_{\partial\Omega}=\gamma(x,y)$$

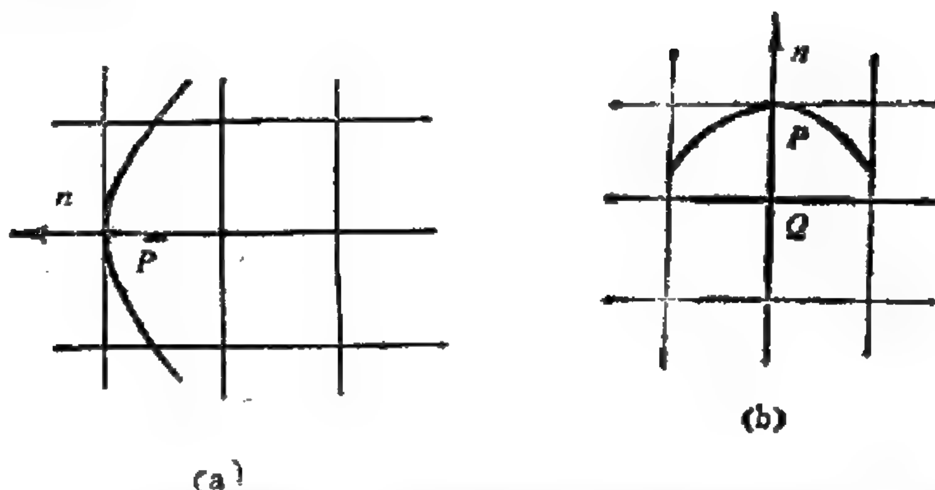
仍假定正方形网格．有以下三种情况：

1° 点P在边界$\partial\Omega$上，且外法向n与坐标轴平行(见图

12.1.18)

$$\frac{u(P)-u(Q)}{h}+ku(P)=\gamma(P)$$

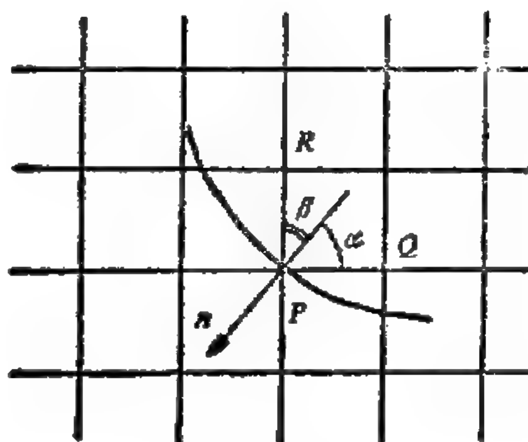
12.1.18 图



(a)

2° 点P在边界$\partial\Omega$上，但外法向n与坐标轴不平行（见图12.1.19）

12.1.19 图



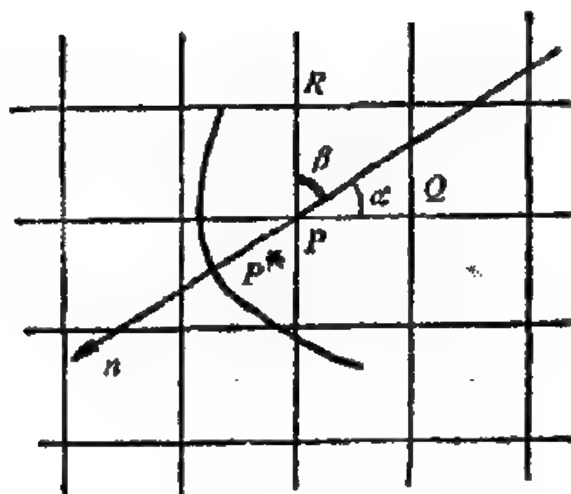
$$\frac{u(P)-u(Q)}{h}\cos\alpha+\frac{u(P)-u(R)}{h}\cos\beta+ku(P)=\gamma(P)$$

3° 点P不在边界$\partial\Omega$上（见图12.1.20）

$$\frac{u(P)-u(Q)}{h}\cos\alpha+\frac{u(P)-u(R)}{h}\cos\beta+ku(P)=\gamma(P^*)$$

其中P^*为距P最近的$\partial\Omega$上一点.

12.1.20 图



12.1.21 **例** 在 $\Omega=\{(x,y)\mid 0<x<0.5,0<y<0.5\}$ 内考虑Laplace方程

$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0$$

边界条件为

$u(0,y)=0, u(x,0)=0, u(x,0.5)=200x, u(0.5,y)=200y$

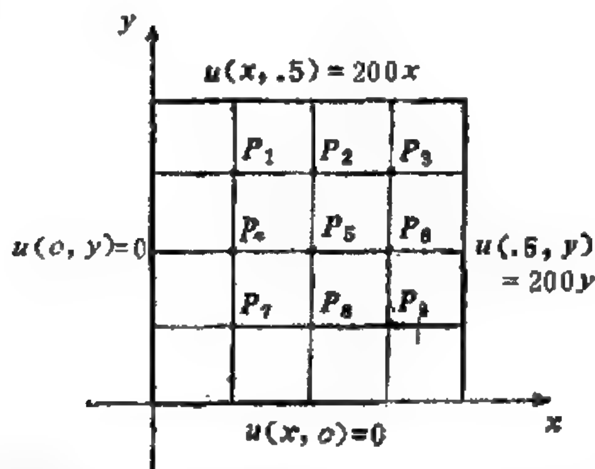
用正方形网格$h=0.125$来求解上述问题。

解 采用差分格式(12.1.10)

$$4u_{ij}-u_{i+1,j}-u_{i-1,j}-u_{i,j+1}-u_{i,j-1}=0$$

$$(i=1,2,3,\ j=1,2,3)$$

12.1.22 图



用网格点来表明这些量(网格点分布见图21.1.22) 则方程可

以写为

$$4u_1 - u_2 - u_4 = u_{0,3} + u_{1,4}$$

$$4u_2 - u_3 - u_1 - u_5 = u_{2,4}$$

$$4u_3 - u_2 - u_6 = u_{4,3} + u_{3,4}$$

$$4u_4 - u_5 - u_1 - u_7 = u_{0,2}$$

$$4u_5 - u_6 - u_4 - u_2 - u_8 = 0$$

$$4u_6 - u_5 - u_3 - u_9 = u_{4,2}$$

$$4u_7 - u_8 - u_4 = u_{0,1} + u_{1,0}$$

$$4u_8 - u_9 - u_7 - u_5 = u_{2,0}$$

$$4u_9 - u_8 - u_6 = u_{3,0} + u_{4,1}$$

方程的右边项可从边界条件的直接转移法得到：

$$u_{1,0} = u_{2,0} = u_{3,0} = u_{0,1} = u_{0,2} = u_{0,3} = 0$$

$$u_{1,4} = u_{4,1} = 25,\quad u_{2,4} = u_{4,2} = 50,\quad u_{3,4} = u_{4,3} = 75$$

从而可得线性方程组

$$\begin{pmatrix} 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 & 0 \\ -1 & 4 & -1 & 0 & -1 & 0 & 0 & 0 & 0 \\ 0 & -1 & 4 & 0 & 0 & -1 & 0 & 0 & 0 \\ -1 & 0 & 0 & 4 & -1 & 0 & -1 & 0 & 0 \\ 0 & -1 & 0 & -1 & 4 & -1 & 0 & -1 & 0 \\ 0 & 0 & -1 & 0 & -1 & 4 & 0 & 0 & -1 \\ 0 & 0 & 0 & -1 & 0 & 0 & 4 & -1 & 0 \\ 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 & -1 \\ 0 & 0 & 0 & 0 & 0 & -1 & 0 & -1 & 4 \end{pmatrix} \begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \end{pmatrix} = \begin{pmatrix} 25 \\ 50 \\ 150 \\ 0 \\ 0 \\ 50 \\ 0 \\ 0 \\ 25 \end{pmatrix}$$

用Gauss－Seidel方法求解上述线性代数方程组，其结果如下：

i	1	2	3	4	5	6	7	8	9
u_i	18.75	37.50	56.25	12.50	25.00	37.50	6.25	12.50	18.75

本例的解析解为$u(x,y)=400xy$，因此求解是精确的。在利用了内点的差分格式和边界条件的处理之后，一般都可得到线性代数方程组，对此可以用Jacobi 方法、Gauss－Seidel 方法、逐次超松弛方法和对称逐次超松弛方法来求解，并且还可利用许多加速收敛的方法.

12.2

12.2 双曲型方程的差分解法

12.2.1 典型问题

最简单的**双曲型方程**（Hyperbolic Equations）是**对流方程**(Convection Equations)．

12.2.1 $\dfrac{\partial u}{\partial t}+a\dfrac{\partial u}{\partial x}=0$ ， $-\infty<x<\infty$，$t\geqslant0$

设给定初始条件

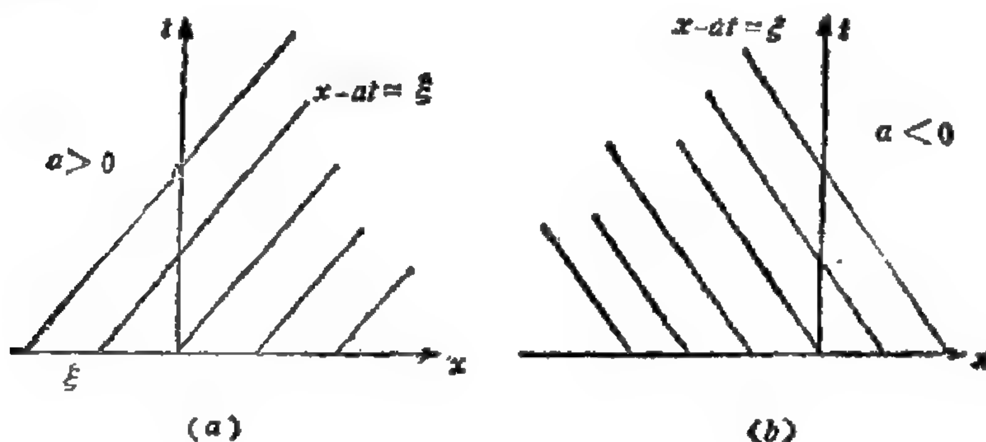
12.2.2 $u(x,0)=\varphi(x)$ ，$-\infty<x<\infty$

容易验证对流方程(12.2.1)的解为

12.2.3 $u(x,t)=\varphi(x-at)$，$-\infty<x<\infty$，$t\geqslant0$

这意味着在$x-t$平面上沿每条直线$x-at=\text{Const}$．u值保持不变，这种直线就是**特征线**（Characteristic Line）（见图12.2.4）。

12.2.4 图



(a) (b)

当$a>0$时，特征线向左倾斜；当$a<0$时，特征线向右倾斜。在任一点(x,t)上的u值仅依赖于初始函数φ在$\xi=x-at$的值，ξ就是通过(x,t)的特征线与x轴交点的横坐标。故解u关于初值的**依赖区域**(Domain of Dependence)是用单个点ξ来表示，如图(12.2.4a)。

如果定解区域$\Omega=\{(x,t)|\ 0\leqslant x\leqslant l, t\geqslant 0\}$，则要给出适当的边界条件。由于对流方程的解在其特征线上是不变的，所以边界条件不能任意给定。如果$a>0$，只能在左边界给条件；如果$a<0$，只能在右边界给条件。

双曲型方程的另一个典型例子是**波动方程**(Wave Equation)

12.2.5 $$\frac{\partial^2 u}{\partial t^2}=a^2\frac{\partial^2 u}{\partial x^2},\quad -\infty<x<\infty,\quad t\geqslant 0$$

其中$a>0$。给定初始条件

12.2.6 $$u(x,0)=\varphi(x),\quad -\infty<x<\infty$$
$$\frac{\partial u(x,0)}{\partial t}=\psi(x),\quad -\infty<x<\infty$$

则初值问题(12.2.5)，(12.2.6)的解可由D'Alembert(达朗贝尔)公式给出：

12.2.7 $$u(x,t)=\frac{\varphi(x+at)+\varphi(x-at)}{2}+\frac{1}{2a}\int_{x-at}^{x+at}\psi(\xi)d\xi$$

由D'Alembert公式不难看出，初值问题(12.2.5)，(12.2.6)的解u在点(x^*,t^*)的值$u(x^*,t^*)$，只依赖于x轴上由x^*-at^*到x^*+at^*之间的初值φ和ψ，因此区间$[x^*-at^*, x^*+at^*]$称为点(x^*,t^*)的**依赖区间**(或称依赖区域)。

对于波动方程(12.2.5)，还有初边值混合问题。设混合问题的求解区域

$\Omega=\{(x,t)|\ 0\leqslant x\leqslant l,\ t\geqslant 0\}$

由此波动方程及其初始条件的空间变量x都限于区间$[0,l]$。

1° 对第一类边界条件，有

$$u|_{x=0}=\Phi_0(t),\quad u|_{x=l}=\Phi_1(t),\quad t\geqslant 0$$

2° 对第三类边界条件，有

$$\left(\frac{\partial u}{\partial x}+\eta_0(t)u\right)\Big|_{x=0}=F_0(t),\quad t\geqslant 0$$

$$\left(\frac{\partial u}{\partial x}+\eta_1(t)u\right)\Big|_{x=l}=F_1(t),\quad t\geqslant 0$$

12.2.2 差分格式

求解对流方程和波动方程的初值问题时，求解区域为

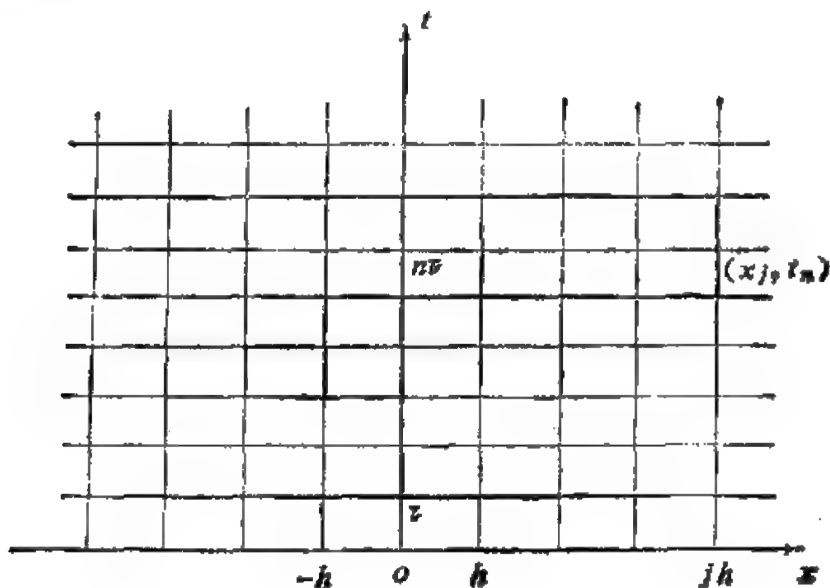
$$\Omega=\{(x,t)|-\infty<x<\infty,\ t\geqslant 0\}$$

即为$x-t$的上半平面。为建立差分方程，首先要剖分网格，设h为x轴方向的步长(空间步长)，τ为t轴方向的步长（时间步长)。令

$$x_j=jh\quad (j=0,\pm1,\pm2,\cdots),\quad t_n=n\tau\quad (n=0,1,2,\cdots)$$

这是二族平行于x轴和t轴的直线，它们构成了$x-t$上半平面的网格，如图(12.2.8)。

12.2.8 图



一般情况下，h可预先给定，τ则根据不同的差分格式而定。

构造差分格式最简单的方法，是用差商代替微商的方法（见12.1）。以对流方程（12.2.1）为例，构造二个常见格式。假定$a>0$，注意到

$$\frac{\partial u(x_j,t_n)}{\partial t}=\frac{u(x_j,t_{n+1})-u(x_j,t_n)}{\tau}+O(\tau)$$

$$\frac{\partial u(x_j,t_n)}{\partial x}=\frac{u(x_{j+1},t_n)-u(x_j,t_n)}{h}+O(h)$$

$$\frac{\partial u(x_j,t_n)}{\partial x}=\frac{u(x_j,t_n)-u(x_{j-1},t_n)}{h}+O(h)$$

这样立即就可得出逼近对流方程（12.2.1）的二个差分格式：

12.2.9 $$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_{j+1}^n-u_j^n}{h}=0$$

12.2.10 $$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_j^n-u_{j-1}^n}{h}=0$$

其中u_j^n是在网格点上取值的函数。

当由第n个时间层（在$t=n\tau$的一排节点）推进到$n+1$层时，公式（12.2.9）和（12.2.10）都提供了逐点计算u_j^{n+1}的明显表达式，因此称它们为**显式格式**(Explicit Scheme)。在公式（12.2.9）中，在推算$n+1$层时只用到第n层的数据，前后联系到二个时间层次，所以格式（12.2.9）称为**两层差分格式**(Two-level Difference Scheme)。同样，差分格式（12.2.10）也是两层差分格式。

给出的差分格式是否逼近到微分方程，以及逼近微分方程的程度如何都可以用截断误差来描述。所谓截断误差就是把微分方程的光滑解代入到差分方程中并作Taylor级数展开后所得到的余项，容易验证，差分方程（12.2.9）和（12.2.10）

的截断误差是$O(\tau)+O(h)$，一般简写为$O(\tau+h)$。如果步长h，$\tau\to0$，截断误差也趋于0，那么称差分方程与微分方程是相容的。**相容性**（Consistency）表示差分方程“收敛”于微分方程，这是用差分方程求解的必备条件。由于差分格式是多种多样的，因此截断误差也将不同。如果截断误差$E=O(\tau^p+h^q)$，则称差分格式对时间（τ）为p阶精度，对空间（h）为q阶精度。如果$p=q$，则称差分格式是p阶精度的，显然差分格式（12.2.9），（12.2.10）都是一阶精度差分格式。

对于一个差分格式，要弄清两个问题：

1° 当步长$h,\tau\to0$时，差分格式的解是否收敛到微分方程的初值问题的解；

2° 计算中产生的误差，在以后的计算中是无限增加还是可以控制。

第一个问题称为**收敛性**（Convergence）问题，第二个问题称为稳定性（Stability）问题。

考虑差分格式(12.2.10)的收敛性问题。先把（12.2.10）变形

$$u_j^n=(1-a\lambda)u_j^{n-1}+a\lambda u_{j-1}^{n-1}$$

其中$\lambda=\tau/h$，一般称其为网格比。

可以看出，u_j^n依赖于$n-1$时间层的u_j^{n-1}，u_{j-1}^{n-1}；而这两个值又依赖于$n-2$时间层的u_j^{n-2}，u_{j-1}^{n-2}，u_{j-2}^{n-2}；依此类推下去可以知道，u_j^n最终依赖于初始层$n=0$上的下列值（初值条件给出）

$$\varphi_{j-n},\varphi_{j-n+1},\cdots,\varphi_j$$

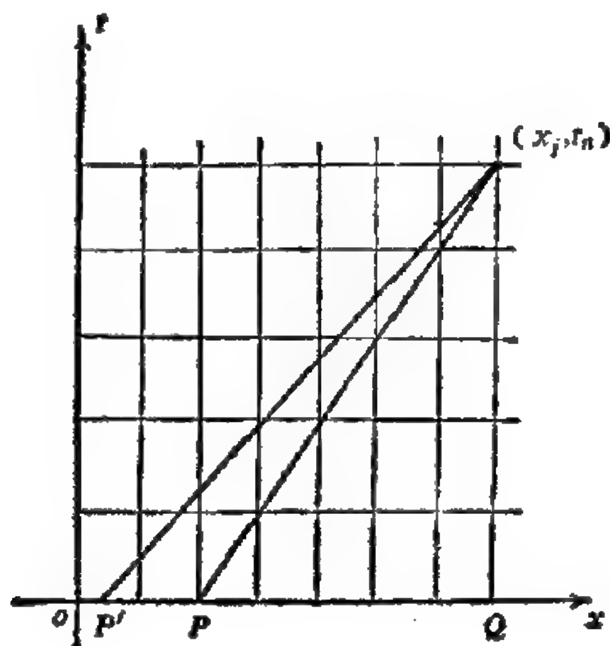
因此，称x轴上含于区间$[x_{j-n},x_j]$上的网格点为差分解u_j^n在点(x_j,t_n)的依赖区域，有时也称区间$[x_{j-n},x_j]$为差分解u_j^n在点（x_j,t_n）的依赖区域。这依赖区域即是过点(x_j,t_n)的两条直线

$$x - x_j = \frac{h}{\tau}(t - t_n) \quad 和 \quad x = x_j$$

在x轴上截出的区间.

如果$a\frac{\tau}{h} > 1$,即$\frac{\tau}{h} > \frac{1}{a}$,那么微分方程的解$u$在点$(x_j,t_n)$的依赖区域$[P',Q]$大于差分解$u_j^n$在点$(x_j,t_n)$的依赖区域$[P,Q]$，见图(12.2.11)．

12.2.11 图



让网格步长变小,但网格比$\frac{\tau}{h}$保持不变,则依赖区域$[P,Q]$和$[P',Q]$不变.显然，若改变(P',P)上的初值，但$[P,Q]$上的初值不变，则微分方程的初值问题的解$u(x_j,t_n)$可取不同的值，而当$h\to 0,\tau\to 0$时（τ/h不变）差分格式的解u_j^n是一串确定的数，它不可能收敛到不同的$u(x_j,t_n)$，亦即当$a\frac{\tau}{h} > 1$时，差分格式不收敛.由此可以得出差分格

式（12.2.10）收敛的必要条件是差分格式的依赖区域包含微分方程初值问题的依赖区域，即

12.2.12 $a\frac{\tau}{h}\leqslant 1$

此条件称为 Courant-Friedrichs-Lewy（库朗-弗兰特里希斯-勒维）条件，也称Courant条件。可以证明，它也是差分格式(12.2.10)收敛的充分条件。类似的推导可以得出差分格式（12.2.9）是不收敛的。

考虑差分格式(12.2.10)的稳定性问题。设初值u_j^0受扰，即$(u+\varepsilon)_j^0=u_j^0+\varepsilon_j^0$；相应地$u_j^n$也受扰，即$(u+\varepsilon)_j^n=u_j^n+\varepsilon_j^n$，容易得出$\varepsilon_j^n$也满足差分方程

12.2.13 $\frac{\varepsilon_j^{n+1}-\varepsilon_j^n}{\tau}+a\frac{\varepsilon_j^n-\varepsilon_{j-1}^n}{h}=0$

把初始误差ε_j^0表示为一个简谐波的形式

$\varepsilon_j^0=(e^{i\omega x})_{x=jh}=e^{i\omega jh}$

其中ω为频率参数，要求形如

12.2.14 $\varepsilon_j^n=[G(\omega)]^n e^{i\omega jh}\quad(j=0,\pm1,\pm2,\cdots)$

的谐波解，其中$G=G(\omega)$为对应于ω的增长因子（Amplification Factor）。将ε_j^n的表达式(12.2.14)代入(12.2.13)，整理得

$$\frac{G-1}{\tau}+a\frac{1-e^{-i\omega h}}{h}=0$$

此方程称为差分方程（12.2.10.）的特征方程，它的根即增长因子

12.2.15 $G=G(\omega)=1-a\lambda(1-e^{-i\omega h})$

其中$\lambda=\tau/h$。

对于(12.2.14)，当$|G(\omega)|>1$时，误差随n作指数状增长；$|G(\omega)|\leqslant 1$时，误差不增长，由于初始误差可以表示为不同

频率ω的谐波的迭加，在计算中舍入误差具有随机性，应该认为所有的ω的频率组分都是可能出现的，因此数值稳定的条件对所有的实数ω都有

12.2.16 $|G(\omega)|\leqslant 1$

对于差分格式（12.2.10），其增长因子为

$$G=1-a\lambda(1-\cos\omega h)-a\lambda i\sin\omega h$$

$$|G|^2=[1-a\lambda(1-\cos\omega h)]^2+a^2\lambda^2\sin^2\omega h$$

$$=1-4a\lambda(1-a\lambda)\sin^2\frac{\omega h}{2}$$

所以，当$a\lambda\leqslant 1$时有$|G|\leqslant 1$，即差分格式(12.2.10)在条件$a\lambda\leqslant 1$之下是稳定的，否则不稳定。

差分格式的稳定性判别条件（12.2.6)是具有一般性的。为了从线性常系数差分方程得到判别稳定性用的特征方程，只要将u_j^n以$G^n e^{i\omega jh}$代入相应的差分方程即可。

对于差分格式（12.2.9），特征方程为

$$\frac{G-1}{\tau}+a\frac{e^{i\omega h}-1}{h}=0$$

从而得增长因子

$$G=1+a\lambda(1-e^{i\omega h})=1+a\lambda(1-\cos\omega h)-a\lambda i\sin\omega h$$

$$|G|^2=[1+a\lambda(1-\cos\omega h)]^2+a^2\lambda^2\sin^2\omega h$$

$$=1+4a\lambda(1+a\lambda)\sin^2\frac{\omega h}{2}$$

因此，不可能选择网格比λ使$|G|\leqslant 1$，所以差分格式是不稳定的。

差分格式(12.2.10)在条件$a\lambda\leqslant 1$之下是稳定的，这种在一定条件下稳定的差分格式称为**条件稳定**的差分格式；而像（12.2.9）那样的差分格式则称为**绝对不稳定**的差分格式。差分格式（12.2.10）在条件$a\lambda\leqslant 1$之下既稳定也收敛，而差

分格式（12.2.9）不但不稳定而且不收敛.稳定性和收敛性的关系由Lax（拉克斯）等价定理给出.

12.2.17 **定理** 给定一个适定的线性初值问题及其相应的相容的差分格式，则差分格式的收敛性等价于差分格式的稳定性.

所谓适定的初值问题，是指该初值问题的解存在、唯一并连续依赖于初始数据.

12.2.3 对流方程的差分格式

对流方程是最简单的双曲型方程，对其差分格式的研究将有助于求解复杂的双曲型方程和双曲型方程组.

逼近对流方程（12.2.9）的中心差分格式

12.2.18 $$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_{j+1}^n-u_{j-1}^n}{2h}=0$$

是很自然的一个差分格式，其截断误差为$O(\tau+h^2)$．容易求出其增长因子

$G(\omega)=1-a\lambda i\sin\omega h$

$|G|^2=1+a^2\lambda^2\sin^2\omega h$

因此这是一个绝对不稳定的格式.

把格式（12.2.18）修改为

12.2.19 $$\frac{u_j^{n+1}-\frac{1}{2}(u_{j+1}^n+u_{j-1}^n)}{\tau}+a\frac{u_{j+1}^n-u_{j-1}^n}{2h}=0$$

称其为 Lax-Friedrichs（拉克斯-弗里特里希斯）格式，容易求出其截断误差$E=O\left(\frac{h^2}{\tau}\right)+O(\tau+h^2)$．由于在双曲型方程的计算中，一般采用$\tau=O(h)$，因而格式（12.2.19）的截断误差化为$E=O(\tau+h)$.这是一个一阶精度的格式，其增长因子

$G(\omega)=\cos\omega h-ia\lambda\sin\omega h$

$|G|^2=1-(1-a^2\lambda^2)\sin^2\omega h$

故格式（12.2.19）的稳定性条件是 $|a|\lambda\leqslant 1$.

为提高差分格式的精度，考虑差分格式

$$\frac{u_j^{n+1}-u_j^{n-1}}{2\tau}+a\frac{u_{j+1}^n-u_{j-1}^n}{2h}=0 \tag{12.2.20}$$

一般称为**蛙跳格式**（Leapfrog Scheme）．显然，这是一个二阶精度格式，其稳定性条件为 $|a|\lambda<1$．要计算 $n+1$ 层上的值 u_j^{n+1}，就必须应用 $n-1, n$ 两个时间层的值 $u_j^{n-1}, u_{j+1}^n, u_{j-1}^n$，前后联系到三个时间层次，因此这样的格式称为**三层格式**（Three-Level Scheme）．这种格式在实际计算时，必须保留二个时间层的数据，从而增加计算机的存贮量；此外，从第0层推进到第1层还必须用另外格式来补充，所以使用中有一定的不便．

从中心差分格式(12.2.18)进行修正，还可以得到另一个很有效的格式．设 $u=u(x,t)$ 是对流方程（12.2.1）的光滑解，把它代入差分格式并进行Taylor级数展开，

$$\frac{u(x_j,t_{n+1})-u(x_j,t_n)}{\tau}+a\frac{u(x_{j+1},t_n)-u(x_{j-1},t_n)}{2h}$$

$$=\left(\frac{\partial u}{\partial t}\right)_j^n+\frac{\tau}{2}\left(\frac{\partial^2 u}{\partial t^2}\right)_j^n+\left(a\frac{\partial u}{\partial x}\right)_j^n+O(\tau^2+h^2)$$

由于 $u=u(x,t)$ 是对流方程的解，所以有

$$\frac{\partial u}{\partial t}=-a\frac{\partial u}{\partial x}$$

$$\frac{\partial^2 u}{\partial t^2}=-a\frac{\partial}{\partial t}\left(\frac{\partial u}{\partial x}\right)=-a\frac{\partial}{\partial x}\left(\frac{\partial u}{\partial t}\right)=a^2\frac{\partial^2 u}{\partial x^2}$$

$$\left(\frac{\tau}{2}\frac{\partial^2 u}{\partial t^2}\right)_j^n=\frac{a^2\tau}{2}\left(\frac{\partial^2 u}{\partial x^2}\right)_j^n$$

$$=\frac{a^2}{2}\frac{\tau}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)+O(\tau h^2)$$

由此得到逼近对流方程（12.2.1）的修正中心差分格式

12.2.21 $$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)-\frac{\tau a^2}{2h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)=0$$

此格式的截断误差 $E=O(\tau^2+h^2)$，所以是二阶精度格式，其稳定性条件是$|a|\lambda\leqslant1$.这是一个二层格式，使用方便，因而受到普遍重视.格式（12.2.21）也称为Lax-Wendroff（拉克斯-温德罗夫）格式，其经常使用的还有以下形式：

$$u_j^{n+1}=u_j^n-\frac{1}{2}a\lambda(u_{j+1}^n-u_{j-1}^n)+\frac{1}{2}a^2\lambda^2(u_{j+1}^n-2u_j^n+u_{j-1}^n)$$

其中$\lambda=\tau/h$.

当$a>0$时，差分格式（12.2.9）是不稳定的，而格式(12.2.10)在条件$a\lambda\leqslant1$之下是稳定的；如果设$a<0$，同样分析可以知道，差分格式(12.2.10)是绝对不稳定的，而格式(12.2.9)在条件$|a|\lambda\leqslant1$之下是稳定的. 所以，差分格式(12.2.9)和(12.2.10)稳定与否同对流方程(12.2.1)中a的符号有关.由于对流方程(12.2.1)有一族特征线

$x-at=\text{const}$

当$a>0$时，特征线向右倾斜；当$a<0$时，特征线向左倾斜，因此所构造的差分格式是否稳定，实质上与特征线走向有关. 如果差分格式与微分方程特征线的走向一致，则在$|a|\lambda\leqslant1$条件下是稳定的；反之，差分格式与微分方程特征线走向不一致，则对任意的网格比都是不稳定的. 沿特征线走向建立的差分格式称为**特征型差分格式**，通常也称**迎风差分格式**（Upwind Difference Scheme）。

利用微分方程的特征线以及微分方程的解在特征线上为常数这一事实，还可以构造出**二阶迎风差分格式**. 设$a>0$，

其格式可以写为

12.2.22 u_j^{n+1}

$$=u_j^n-a\lambda(u_j^n-u_{j-1}^n)-\frac{a\lambda}{2}(1-a\lambda)(u_j^n-2u_{j-1}^n+u_{j-2}^n)$$

这个格式的稳定性条件是 $a\lambda\leqslant 2$。

逼近对流方程(12.2.1)的差分格式还可以是

12.2.23 $$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_j^{n+1}-u_{j-1}^{n+1}}{h}=0 \quad (a>0)$$

这个格式在第 $(n+1)$ 时间层上有多于一个点上的未知函数值出现，故称其为**隐式格式** (Implicit Diffence Scheme)。隐式差分格式适用于求解初边值混合问题或具有周期条件的初值问题.对于隐式格式,一般都要求解代数方程组（但很多双曲型方程的隐式格式则可直接计算).差分格式(12.2.23)的增长因子

$$G=\frac{1}{1+a\lambda(1-e^{-i\sigma h})}$$

所以对任 λ 有 $|G|\leqslant 1$，这样的格式称为**绝对稳定的**.

用隐式差分格式(12.2.23)解初边值混合问题.

设求解区域 $\Omega=\{(x,t)|\ 0\leqslant x\leqslant l,t\geqslant 0\}$，此时网格由直线族

$$x=x_j=jh, \quad h=\frac{l}{J}, \ j=0,1,\cdots,J$$

$$t=t_n=n\tau, \quad \tau>0, \ n=0,1,\cdots$$

组成. 设 $a>0$，则 u_0^{n+1}，$n\geqslant 0$,给定.再由方程(12.2.23)，即

$$u_j^{n+1}=\frac{1}{1+a\lambda}u_j^n+\frac{a\lambda}{1+a\lambda}u_{j-1}^{n+1}, \qquad j=1,2,\cdots,J-1$$

及初始条件 $u_j^0=\varphi_j, j=0,1,\cdots,J$，可以显式（不用解方程

组）地求解，所以这是相当好的。

逼近对流方程(12.2.1)的常见的差分格式如表（12.2.24）所示，其中 $a>0$。

12.2.24 表

名 称	格式与截断误差	稳定条件
迎风格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_j^n-u_{j-1}^n)=0$ $E=O(\tau+h)$	$a\lambda\leqslant 1$
右偏显式格式	$\frac{1}{\tau}(u_j^{n+1}+u_j^n)+\frac{a}{h}(u_{j+1}^n-u_j^n)=0$ $E=O(\tau+h)$	绝对不稳定
迎风隐式格式	$\frac{1}{\tau}(u_j^n-u_j^n)+\frac{a}{h}(u_j^{n+1}-u_{j-1}^{n+1})=0$ $E=O(\tau+h)$	绝对稳定
中心显式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)=0$ $E=O(\tau+h^2)$	绝对不稳定
中心隐式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})=0$ $E=O(\tau+h^2)$	绝对稳定
平均隐式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{1}{2}\left[\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)\right.$ $\left.+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})\right]=0$ $E=O(\tau^2+h^2)$	绝对稳定
Lax-Friedrichs格式	$\frac{1}{\tau}\left[u_j^{n+1}-\frac{1}{2}(u_{j+1}^n+u_{j-1}^n)\right]$ $+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)=0$ $E=O(h^2/\tau)+O(\tau+h^2)$	$a\lambda\leqslant 1$

名　称	格式与截断误差	稳定条件
Lax-Wendroff 格式	$u_j^{n+1}=u_j^n-\frac{1}{2}a\lambda(u_{j+1}^n-u_{j-1}^n)$ $+\frac{1}{2}a^2\lambda^2(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau^2+h^2)$	$a\lambda\leqslant 1$
蛙跳格式	$\frac{u_j^{n+1}-u_j^{n-1}}{2\tau}+a\frac{u_{j+1}^n-u_{j-1}^n}{2h}=0$ $E=O(\tau^2+h^2)$	$a\lambda\leqslant 1$
二阶迎风格式	$u_j^{n+1}=u_j^n-a\lambda(u_j^n-u_{j-1}^n)$ $-\frac{a\lambda}{2}(1-a\lambda)(u_j^n-2u_{j-1}^n+u_{j-2}^n)$ $E=O(\tau^2+h^2)$	$a\lambda\leqslant 2$
Wendroff 隐式格式	$u_{j+1}^{n+1}=u_j^n+\frac{1-a\lambda}{1+a\lambda}(u_{j+1}^n-u_j^{n+1})$ $E=O(\tau^2+h^2)$	绝对稳定
Roberts-Weiss 隐式格式	$u_j^{n+1}=u_j^n+\frac{a\lambda}{2+a\lambda}(u_{j-1}^{n+1}-u_{j+1}^n)$ $E=O(\tau^2+h)$	绝对稳定

12.2.4 波动方程的差分格式

波动方程（12.2.5）的差分格式可以直接由差商代替微商而得到

12.2.25 $$\frac{u_j^{n+1}-2u_j^n+u_j^{n-1}}{\tau^2}=a^2\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}$$

这是一个二阶精度的显式格式，其稳定性条件是 $a\lambda<1$，其中 $\lambda=\tau/h$。隐式格式的形式很多，差分格式

12.2.26 $$\frac{u_j^{n+1}-2u_j^n+u_j^{n-1}}{\tau^2}=a^2\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}$$

是一个隐式格式，其截断误差为$O(\tau+h^2)$，这是一个绝对稳定的差分格式。更一般的差分格式是von Neumann（冯诺依曼）格式

$$\frac{u_j^{n+1}-2u_j^n+u_j^{n-1}}{\tau^2}=a^2\left\{\theta\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}\right. \tag{12.2.27}$$

$$\left.+(1-2\theta)\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}+\theta\frac{u_{j+1}^{n-1}-2u_j^{n-1}+u_{j-1}^{n-1}}{h^2}\right\}$$

其中$0\leqslant\theta\leqslant1$。利用Taylor级数展开，可以直接验证对任$\theta\in[0,1]$，截断误差都是$O(\tau^2+h^2)$。此格式的稳定性条件是：如果$\frac{1}{4}\leqslant\theta\leqslant1$，则差分格式绝对稳定；如果$0\leqslant\theta<\frac{1}{4}$，则差分格式的稳定性条件为$0<a\lambda<\frac{1}{\sqrt{1-4\theta}}$。特别地，当$\theta=0$时，差分格式（12.2.27）化为显式格式。

格式（12.2.27）的另外两个重要的特殊情形是$\theta=\frac{1}{2}$和$\theta=\frac{1}{4}$。当$\theta=\frac{1}{2}$时，差分格式化为

$$\frac{u_j^{n+1}-2u_j^n+u_j^{n-1}}{\tau^2}$$

$$=a^2\frac{1}{2}\left[\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}+\frac{u_{j+1}^{n-1}-2u_j^{n-1}+u_{j-1}^{n-1}}{h^2}\right]$$

此格式称为**平均隐式格式**，由于$\theta\geqslant\frac{1}{4}$，所以是稳定的。$\theta=\frac{1}{4}$在实际应用中是很受重视的一个差分格式。

关于初始条件的差分近似。对于第一个初始条件$u(x,0)=\varphi(x)$，$-\infty<x<\infty$，可以用直接转移的方法，得到

12.2.28 $u_j^0=\varphi(jh)=\varphi_j \quad (j=0,\pm1,\pm2,\cdots)$

对于第二个初始条件$\frac{\partial u(x,0)}{\partial t}=\psi(x) \quad (-\infty<x<\infty)$，一种方法是

12.2.29 $\frac{u_j^1-u_j^0}{\tau}=\psi(jh)=\psi_j \quad (j=0,\pm1,\pm2,\cdots)$

容易看出，(12.2.29)的截断误差为$O(\tau)$，如果采用von Neumann差分格式（包括$\theta=0$，$\frac{1}{4}$，$\frac{1}{2}$)进行计算，那么初始条件的差分近似与方程的差分近似不适应。为提高（12.2.29）的截断误差可采用另一种方法，即

12.2.30 $\frac{u_j^1-u_j^{-1}}{2\tau}=\psi_j \quad (j=0,\pm1,\pm2,\cdots)$

这个方法的截断误差为$O(\tau^2)$。但是，因为它又引进了新的未知数u_j^{-1}，所以由（12.2.30）不能确定u_j^1。为确定u_j^1，应把（12.2.30）和在$(x_j,0)$上的差分方程联立求出u_j^1。设采用显式格式进行计算，那么在$(x_j,0)$上有

$$\frac{u_j^1-2u_j^0+u_j^{-1}}{\tau^2}=a^2\frac{u_{j+1}^0-2u_j^0+u_{j-1}^0}{h^2}$$

此式与（12.2.30）联立，可以消去u_j^{-1}并得到

12.2.31 $u_j^1=\frac{a^2\lambda^2}{2}(\varphi_{j-1}+\varphi_{j+1})+(1-a^2\lambda^2)\varphi_j+\tau\psi_j$

其中$\lambda=\tau/h$。

有了初始条件的离散，利用（12.2.28）和（12.2.29）（或12.2.31）可以算出初始层$(n=0)$及第一层$(n=1)$各网格点上的值，然后再利用差分格式逐层算出任意网格点上的值。

关于波动方程的初边值混合问题．由求解区域$\Omega=$

$\{(x,t)|\ 0\leqslant x\leqslant l,t\geqslant 0\}$，因此网格由直线

$$x=x_j=jh,\ h=\frac{l}{J},\quad j=0,1,2,\cdots,J$$

$$t=t_n=n\tau,\ \tau>0,\ n\geqslant 0$$

所构成。对于边界条件的离散，有以下方法：

1° 第一类边界条件采用直接转移，

12.2.32 $\quad u_0^n=\Phi_0^n=\Phi_0(n\tau),\ u_J^n=\Phi_1^n=\Phi_1(n\tau)$

2° 第三类边界条件采用两种办法。第一种办法是直接用差商代替微商

12.2.33
$$\begin{cases}\dfrac{u_1^n-u_0^n}{h}+\eta_0^n u_0^n=F_0^n\\[2ex]\dfrac{u_J^n-u_{J-1}^n}{h}+\eta_1^n u_J^n=F_1^n\end{cases}$$

其中

$\eta_0^n=\eta_0(n\tau),\ \eta_1^n=\eta_1(n\tau),\ F_0^n=F_0(n\tau),F_1=F_1(n\tau)$

其截断误差为 $O(h)$。第二种办法是将网格平移半个步长，即在网格

12.2.34
$$\begin{cases}x_j=(j+\dfrac{1}{2})h,\quad h=\dfrac{l}{J},\ j=-1,0,1,\cdots,J\\ t_n=n\tau,\ n\geqslant 0\end{cases}$$

上讨论差分近似。此情况下第三边界条件的差分近似是

12.2.35
$$\begin{cases}\dfrac{u_0^n-u_{-1}^n}{h}+\eta_0^n\dfrac{u_0^n+u_{-1}^n}{2}=F_0^n\\[2ex]\dfrac{u_J^n-u_{J-1}^n}{h}+\eta_1^n\dfrac{u_J^n+u_{J-1}^n}{2}=F_1^n\end{cases}$$

其截断误差为 $O(h^2)$。利用这一差分近似时，在算出所有的 u_j^n 在内部节点上的值以后，应再用插值法求出 $x=0$ 和 $x=l$ 上 u 的值。

12.3 抛物型方程的差分解法

12.3.1 典型问题

抛物型方程（Parabolic Equations）的最简单的方程是**扩散方程**（Diffusion Equation）

12.3.1 $$\frac{\partial u}{\partial t}=b\frac{\partial^2 u}{\partial x^2}\quad(-\infty<x<\infty,\ t\geqslant 0)$$

其中 $b>0$. 此方程也称热传导方程，如果给定初始条件

12.3.2 $$u(x,0)=\varphi(x)\quad(-\infty<x<\infty)$$

则就构成了初值问题。这个初值问题的解可以表为

12.3.3 $$u(x,t)=\frac{1}{\sqrt{4\pi bt}}\int_{-\infty}^{\infty}\exp\left[-\frac{(x-\xi)^2}{4bt}\right]\varphi(\xi)d\xi$$

$$(-\infty<x<\infty,t>0)$$

对于扩散方程(12.3.1)还有初边值混合问题，即在区域 $\Omega=\{(x,t)\mid 0\leqslant x\leqslant l,t\geqslant 0\}$ 内考虑扩散方程的求解，除了给出在 Ω 内的方程（12.3.1）和初始条件（12.3.2）之外，还要给出边界条件. 边界条件提法如下：

1° 第一类边界条件

12.3.4 $$\begin{cases}u(0,t)=\mu_1(t), & t\geqslant 0\\ u(l,t)=\mu_2(t), & t\geqslant 0\end{cases}$$

2° 第三类边界条件

12.3.5 $$\begin{cases}\left(\dfrac{\partial u}{\partial x}-\lambda_1(t)u\right)\Big|_{x=0}=\nu_1(t), & t\geqslant 0\\ \left(\dfrac{\partial u}{\partial x}+\lambda_2(t)u\right)\Big|_{x=l}=\nu_2(t), & t\geqslant 0\end{cases}$$

其中 $\lambda_i(t)\geqslant 0,i=1,2$. 如果在（12.3.5）中，$\lambda_1,\lambda_2$ 为零，则称为第二类边界条件。

抛物型方程的另一典型例子是**对流扩散方程**（Convection Diffusion Equation）

12.3.6 $$\frac{\partial u}{\partial t}+a\frac{\partial u}{\partial x}=b\frac{\partial^2 u}{\partial x}\quad(-\infty<x<\infty,\ t\geqslant 0)$$

其中 $b>0$. 这个方程可以看成是对流方程和扩散方程的耦合，当 b 很小时，方程反映对流方程的特征，当 a 很小时，方程反映扩散方程的特征. 如果仍取（12.3.2）作为初始条件，则构成对流扩散方程的初值问题，这个初值问题的解可以表为

12.3.7 $$u(x,t)=\frac{1}{\sqrt{4\pi bt}}\int_{-\infty}^{\infty}\exp\left[-\frac{(x-\xi-at)^2}{4bt}\right]\varphi(\xi)d\xi$$

最简单的二维扩散方程的初边值问题为：

12.3.8 $$\begin{cases}\dfrac{\partial u}{\partial t}=b\left(\dfrac{\partial^2 u}{\partial x^2}+\dfrac{\partial^2 u}{\partial y^2}\right),\quad 0<x<l,\ 0<y<l\\ \qquad\qquad t>0,\ b>0\\ u(x,y,0)=\varphi(x,y),\quad 0\leqslant x,\ y\leqslant l\\ u(0,y,t)=\mu_1(y,t),\ u(l,y,t)=\mu_2(y,t)\\ \qquad\qquad 0\leqslant y\leqslant l,\ t\geqslant 0\\ u(x,0,t)=\nu_1(x,t),u(x,l,t)=\nu_2(x,t),0\leqslant x\leqslant l,t\geqslant 0\end{cases}$$

12.3.2 扩散方程的差分格式

对于初值问题来说，网格的剖分如下：令 h 和 τ 分别是 x 轴方向和 t 轴正方向的步长，$x_j=jh,j=0,\pm1,\cdots;t_n=n\tau$，$n=0,1,\cdots$，两组平行线构成了 x-t 上半平面的网格. 利用差商替代微商的办法可以获得一系列逼近扩散方程（12.3.1）的差分格式.

1° 显式格式

12.3.9 $$\frac{u_j^{n+1}-u_j^n}{\tau}=b\,\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}$$

其截断误差为$O(\tau+h^2)$，增长因子

$$G=1-4b\lambda\sin^2\frac{\omega h}{2}$$

其中 $\lambda=\tau/h^2$，称 λ 为 网格比。但是，此处与双曲型方程的差分格式中的网格比含义不同。由增长因子 G，可以得格式（12.3.9）的稳定性条件为

$$b\lambda\leqslant\frac{1}{2}，\text{即}\ b\frac{\tau}{h^2}\leqslant\frac{1}{2}.$$

2° 隐式格式

12.3.10 $$\frac{u_j^{n+1}-u_j^n}{\tau}=b\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}$$

其截断误差为$O(\tau+h^2)$，增长因子

$$G=\frac{1}{1+4b\lambda\sin^2\frac{\omega h}{2}}$$

由此可知，此格式对任何网格比都是稳定的。

3° Crank-Nicolson 格式

12.3.11 $$\frac{u_j^{n+1}-u_j^n}{\tau}=b\frac{1}{2}\left\{\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}+\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}\right\}$$

这是一个绝对稳定的隐式格式，特别是它的截断误差为$O(\tau^2+h^2)$，即为二阶精度格式.这个格式又称为Crank-Nicolson（克兰克-尼科尔松）格式，简记为 C-N 格式。

4° Richardson 格式

12.3.12 $$\frac{u_j^{n+1}-u_j^{n-1}}{2\tau}=b\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}$$

这是一个二阶精度的三层格式，其增长因子

$$G_{1,2}=-4b\lambda\sin^2\frac{\omega h}{2}\pm\sqrt{1+\left(4b\lambda\sin^2\frac{\omega h}{2}\right)^2}$$

因此Richardson（李查逊）**格式**是一个绝对不稳定的格式。

5° Du Fort-Frankel 格式

12.3.13 $$\frac{u_j^{n+1}-u_j^{n-1}}{2\tau}=b\,\frac{u_{j+1}^n-(u_j^{n+1}+u_j^{n-1})+u_{j-1}^n}{h^2}$$

Du Fort-Frankel（**杜福尔-佛兰克尔**）**格式**是Richardson格式的一种修正，是绝对稳定的差分格式。其截断误差

$$E=b\left(\frac{\tau}{h}\right)^2\left(\frac{\partial^2u}{\partial t^2}\right)_j^n+O(\tau^2+h^2)+O\left(\frac{\tau^4}{h^2}\right)$$

如果取$\tau=O(h)$，则当$\tau,h\to0$时E并不趋于0，所以差分格式（12.3.13）与微分方程（12.3.1）是不相容的.只有当τ,h趋于0时$\frac{\tau}{h}$也趋于0，差分格式（12.3.13）才能相容于扩散方程（12.3.1）.

逼近扩散方程（12.3.1)的常见的差分格式如表（12.3.14）所示

12.3.14 表

名称	差分格式与截断误差	稳定条件
显式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)=b\,\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau+h^2)$	$b\lambda\leqslant\frac{1}{2}$
隐式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)$ $=b\,\frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})$ $E=O(\tau+h^2)$	绝对稳定

名称	差分格式与截断误差	稳定条件
Crank —Nicolson 格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)=b\frac{1}{2h^2}[(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})+(u_{j+1}^n-2u_j^n+u_{j-1}^n)]$ $E=O(\tau^2+h^2)$	绝对稳定
加权隐式 格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)=b\frac{1}{h^2}[\theta(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})+(1-\theta)(u_{j+1}^n-2u_j^n+u_{j-1}^n)]$ $E=b\ (\frac{1}{2}-\theta)\cdot O(\tau)+O(\tau^2+h^2)$	$\frac{1}{2}\leqslant\theta\leqslant1$， 绝对稳定 $0\leqslant\theta<\frac{1}{2}$， $b\lambda\leqslant\frac{1}{2(1-2\theta)}$
	$\theta\geqslant0$； $(1+\theta)\frac{u_j^{n+1}-u_j^n}{\tau}-\theta\frac{u_j^n-u_j^{n-1}}{\tau}$ $=b\ \frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})$ $E=O(\tau+h^2)$	绝对稳定
	$\theta=\frac{1}{2}$； $\frac{3}{2}\frac{u_j^{n+1}-u_j^n}{\tau}-\frac{1}{2}\frac{u_j^n-u_j^{n-1}}{\tau}$ $=b\frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})$ $E=O(\tau^2+h^2)$	绝对稳定

续表

名称	差分格式与截断误差	稳定条件
Du Fort—Frankel 格式	$\frac{u_j^{n+1}-u_j^{n-1}}{2\tau}=b\frac{u_{j+1}^n-(u_j^{n-1}+u_j^{n+1})+u_{j-1}^n}{h^2}$ $E=O(\tau^2+h^2)+O\left(\frac{\tau^2}{h^2}\right)$	绝对稳定
	$\frac{1}{12\tau}(u_{j+1}^{n+1}-u_{j+1}^n)+\frac{5}{6\tau}(u_j^{n+1}-u_j^n)+\frac{1}{12\tau}(u_{j-1}^{n+1}-u_{j-1}^n)$ $=b\frac{1}{2}\left[\frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})+\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)\right]$ $E=O(\tau^2+h^4)$	绝对稳定
Richardson 格式	$\frac{1}{2\tau}(u_j^{n+1}-u_j^{n-1})=b\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau^2+h^2)$	绝对不稳定

对于扩散方程初值问题（12.3.1）和（12.3.2）的求解，还必须有初始条件（12.3.2）的离散，这可以用直接转移法得到

12.3.15 $u_j^0=\varphi(jh)=\varphi_j$

再由(12.3.1)的差分方程及(12.3.15)进行逐层计算。

关于初边值混合问题的求解，网格不同于初值问题。此时设

$$x_j=jh,\ j=0,1,\cdots,J,\quad h=\frac{l}{J}$$

$t_n=n\tau,n=0,1,2,\cdots,\tau$为时间步长

对于第一类边界条件，可以用直接转移法来进行，条件

(12.3.4)的离散为

12.3.16 $u_0^n=\mu_1(n\tau)$, $u_J^n=\mu_2(n\tau)$, $n\geqslant 0$

由于第三类边界条件中含有导数，所以不能用直接转移的办法，而通常采用两种方法(类同于对波动方程的处理)：

1° 第一种方法

12.3.17
$$\begin{cases}\dfrac{u_1^n-u_0^n}{h}-\lambda_1(n\tau)u_0^n=\nu_1(n\tau)\\[2ex] \dfrac{u_J^n-u_{J-1}^n}{h}+\lambda_2(n\tau)u_J^n=\nu_2(n\tau)\end{cases}$$

2° 第二种方法

将网格平移半个步长，即在网格

$$\begin{cases}x_j=(j+\dfrac{1}{2})h, \quad h=\dfrac{l}{J}, \quad j=-1,0,1,\cdots,J\\ t_n=n\tau, \quad n=0,1,2,\cdots\end{cases}$$

上讨论差分近似.在此情况下，第三类边界条件的差分近似为

12.3.18
$$\begin{cases}\dfrac{u_0^n-u_{-1}^n}{h}-\lambda_1(n\tau)\dfrac{u_0^n+u_{-1}^n}{2}=\nu_1(n\tau)\\[2ex] \dfrac{u_J^n-u_{J-1}^n}{h}+\lambda_2(n\tau)\dfrac{u_J^n+u_{J-1}^n}{2}=\nu_2(n\tau)\end{cases}$$

在计算出所有的 u_j^n 在 内 部节点上的值之后，再用插值线求出 $x=0$ 和 $x=l$ 上的值。

12.3.19 **例** 考虑扩散方程的初边值混合问题

$$\begin{cases}\dfrac{\partial u}{\partial t}=\dfrac{\partial^2 u}{\partial x^2}, \quad 0>x>1, \quad t>0\\ u(0,t)=u(1,t)=0, \quad t>0\\ u(x,0)=\sin(\pi x), \quad 0\leqslant x\leqslant 1\end{cases}$$

分别用隐式格式，Crank-Nicolson 格 式 和显式格式解之，并比较其结果。

解 取空间步长 $h=0.1$，时间步长 $\tau=0.01$,网格比 $\lambda=$

$\frac{\tau}{h^2}=1$。

1° 隐式格式

$$\begin{cases}\dfrac{u_j^{n+1}-u_j^n}{\tau}=\dfrac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2} & (j=1,2,\cdots,9)\\ u_0^{n+1}=u_{10}^{n+1}=0\\ u_j^0=\sin(\pi x_j) & (j=0,1,2,\cdots,9,10)\end{cases}$$

考虑到 $\lambda=\tau/h^2=1$，可以把差分格式写为

$$\begin{cases}-u_{j-1}^{n+1}+3u_j^{n+1}-u_{j+1}^{n+1}=u_j^n & (j=1,2,\cdots,9)\\ u_0^{n+1}=u_{10}^{n+1}=0\\ u_j^0=\sin(\pi x_j) & (j=0,1,\cdots,9,10)\end{cases}$$

写成矩阵形式，有

$$\begin{pmatrix}3 & -1 & 0 & \cdots & 0\\ -1 & 3 & -1 & \ddots & \vdots\\ 0 & \ddots & \ddots & \ddots & \vdots\\ \vdots & \ddots & \ddots & \ddots & 0\\ \vdots & & \ddots & \ddots & -1\\ 0 & \cdots & 0 & -1 & 3\end{pmatrix}\begin{pmatrix}u_1^{n+1}\\ u_2^{n+1}\\ \vdots\\ \vdots\\ u_8^{n+1}\\ u_9^{n+1}\end{pmatrix}=\begin{pmatrix}u_1^n\\ u_2^n\\ \vdots\\ \vdots\\ u_8^n\\ u_9^n\end{pmatrix}$$

这是以对角占优的三对角线矩阵为系数矩阵的线性代数方程组，可以用追赶法解之。重复计算，可以得到$t=0.5$的一组解，其结果见表(12.3.20)。

2° Crank-Nicolson 格式

$$\frac{u_j^{n+1}-u_j^n}{\tau}=\frac{1}{2}\left[\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2}+\frac{u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1}}{h^2}\right]\quad(j=1,2,\cdots,9)$$

由于 $\lambda=\tau/h^2=1$，所以上述方程组化为

$$-u_{j-1}^{n+1}+4u_j^{n+1}-u_{j+1}^{n+1}=u_{j+1}^n+u_{j-1}^n \quad (j=1,2,\cdots,9)$$

此仍是以对角占优的三对角线矩阵为系数 的 线 性 代数方程组，可用追赶法解之。重复计算，可以得 到 $t=0.5$ 的 一 组解，其结果见表（12.3.20）。

3° 显式格式

$$u_j^{n+1}=u_{j-1}^n-u_j^n+u_{j+1}^n \quad (j=1,2,\cdots,n)$$

由初始条件开始，直接计算 可 得 到 $t=0.5$ 的一组解，其结果见表（12.3.20）。

本题的解析解为

$$u(x,t)=e^{-\pi^2 t}\sin(\pi x)$$

为比较起见，其在网格点的值也列在表（12.3.20）中。

12.3.20 表

x_j	解析解 $u(x_j,0.5)$	全隐式 u_j^{50}	显式 u_j^{50}	C－N格式 u_j^{50}
0	0	0	0	0
0.1	0.00222241	0.00289802	8.19876×10^7	0.00230512
0.2	0.00422728	0.00551236	-1.55719×10^8	0.00438461
0.3	0.00581836	0.00758711	2.13833×10^8	0.00603489
0.4	0.00683989	0.00891918	-2.50642×10^8	0.0070944
0.5	0.00719188	0.00937818	2.62685×10^8	0.00745954
0.6	0.00683989	0.00891918	-2.49015×10^8	0.0070944
0.7	0.00581836	0.00758711	2.11200×10^8	0.00603489
0.8	0.00422728	0.00551236	-1.53086×10^8	0.00438461
0.9	0.00222241	0.00289802	8.03604×10^7	0.00230512
1.0	0	0	0	0

由表可以看出，Crank-Nicolson格式的结果较为精确，而显式格式的结果已面貌皆非了。这是因为$\lambda=1$，格式已不稳定，所以计算不出正确的数值结果。如果对于显式格式取

$\lambda=0.05$即取 $\tau=0.0005$，此时可得出与 Crank-Nicolson 相近的结果。

所以，在扩散方程的求解中，Crank-Nicolson 格式是十分可取的，而显式格式则一般不宜采用。

12.3.3 对流扩散方程的差分格式

把对流方程的差分格式和扩散方程的差分格式结合起来，容易构造出对流扩散方程的差分格式。

逼近对流扩散方程（12.3.6）的中心显式格式为

$$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_{j+1}^n-u_{j-1}^n}{2h}=b\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2} \tag{12.3.21}$$

其截断误差为$O(\tau+h^2)$，容易求出它的增长因子

$G=[1-2\beta(1-\cos\omega h)]-i\alpha\sin\omega h$

其中

$\alpha=a\dfrac{\tau}{h}$ ， $\beta=b\dfrac{\tau}{h^2}$

由此容易得到差分格式（12.3.21）的稳定性条件是

$\beta\leqslant\dfrac{1}{2}$ 和 $\alpha^2\leqslant2\beta$

为确定起见，令 $a>0$，则逼近对流扩散方程(12.3.6)的迎风显式格式为

$$\frac{u_j^{n+1}-u_j^n}{\tau}+a\frac{u_j^n-u_{j-1}^n}{h}=b\frac{u_{j+1}^n-2u_j^n+u_{j-1}^n}{h^2} \tag{12.3.22}$$

其截断误差为$O(\tau+h)$，增长因子

$G=[1-(2\beta+\alpha)(1-\cos\omega h)]-\alpha i\sin\omega h$

由此可以推出差分格式（12.3.22）的稳定条件为

$\alpha+2\beta\leqslant1$

常用的逼近对流扩散方程(12.3.6)的差分格式列在表(12.3.23)中.在实际计算时，当对流项占优势时，一般宜采用迎风格式。

12.3.23 表（取 $a>0$）

名称	差分格式与截断误差	稳定条件
中心显式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)$ $=b\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau+h^2)$	$\beta\leqslant\frac{1}{2}$ $\alpha^2\leqslant 2\beta$
迎风显式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_j^n-u_{j-1}^n)$ $=b\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau+h)$	$\alpha+2\beta\leqslant 1$
最优显式格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)$ $=(b+\frac{a^2}{2}\tau)\frac{1}{h^2}(u_{j+1}^n-2u_j^n+u_{j-1}^n)$ $E=O(\tau+h^2)$	$\alpha^2+2\beta\leqslant 1$
Du Fort —Frankel 显式格式	$\frac{1}{2\tau}(u_j^{n+1}-u_j^{n-1})+\frac{a}{2h}(u_{j+1}^n-u_{j-1}^n)$ $=b\frac{1}{h^2}[u_{j+1}^n-(u_j^{n+1}+u_j^{n-1})+u_{j-1}^n]$ $E=O(\tau^2/h^2)+O(\tau^2+h^2)$	$\alpha\leqslant 1$
全隐格式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}(u_{j+1}^{n+1}-u_{j-1}^{n+1})$ $=b\frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})$ $E=O(\tau+h^2)$	绝对稳定

续表

名称	差分格式与截断误差	稳定条件
平均隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{2h}[(u_{j+1}^{n+1}-u_{j-1}^{n+1})+(u_{j+1}^n-u_{j-1}^n)]$ $=\frac{b}{2h^2}[(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})+(u_{j+1}^n-2u_j^n+u_{j-1}^n)]$ $E=O(\tau^2+h^2)$	绝对稳定
迎风隐式	$\frac{1}{\tau}(u_j^{n+1}-u_j^n)+\frac{a}{h}(u_j^{n+1}-u_{j-1}^{n+1})$ $=b\frac{1}{h^2}(u_{j+1}^{n+1}-2u_j^{n+1}+u_{j-1}^{n+1})$ $E=O(\tau+h)$	绝对稳定

12.3.4 二维扩散方程

考虑二维扩散方程的初边值混合问题（12.3.8）的差分格式。首先对区域

$\Omega=\{(x,y,t)|\ 0\leqslant x,y\leqslant l,t\geqslant 0\}$

进行网格剖分。为简单起见，x 方向和 y 方向的空间步长皆取 h，时间步长为 τ，这样网格节点可以记为$x_i=ih,y_j=jh$，$t_n=n\tau$，其中 $i,j=0,1,\cdots,M,Mh=l,n\geqslant 0$。引进记号

$\delta_x^2 u_{ij}^n=u_{i+1,j}^n-2u_{ij}^n+u_{i-1,j}^n$

$\delta_y^2 u_{ij}^n=u_{i,j+1}^n-2u_{ij}^n+u_{i,j-1}^n$

其中 u_{ij} 是取在网格点上的函数。

利用差商替代微商的办法，可以直接写出问题(12.3.8)中扩散方程的一些差分格式。一维扩散方程显式格式的直接推广是

12.3.24 $\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\tau}=\frac{b}{h^2}(\delta_x^2 u_{ij}^n+\delta_y^2 u_{ij}^n)$

容易求得此差分格式的截断误差同一维情况一样，即为 $O(\tau+h^2)$。仿一维的方法，可以算出（12.3.24）的增长因子

$$G=1-4b\frac{\tau}{h^2}\sin^2\frac{\omega_1 h}{2}-4b\frac{\tau}{h^2}\sin^2\frac{\omega_2 h}{2}$$

所以差分格式（12.3.24）的稳定性条件是

$$b\frac{\tau}{h^2}\leqslant\frac{1}{4}$$

在一维中，稳定性要求 $b\frac{\tau}{h^2}\leqslant\frac{1}{2}$，由例（21.3.19)可以看出，显式格式的时间步长 τ 是 Crank-Nicolson 格式的时间步长的$\frac{1}{20}$时才能计算出相近的数值结果，而二维扩散方程的显式格式的步长要求更严，所以在实际计算中采用显式格式是不可取的。

一维全隐格式和 Crank- Nicolson 格式的直接推广为

12.3.25 $\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\tau}=\frac{b}{h^2}(\delta_x^2 u_{ij}^{n+1}+\delta_y^2 u_{ij}^{n+1})$

和

12.3.26 $\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\tau}$

$$=\frac{b}{2h^2}(\delta_x^2 u_{ij}^{n+1}+\delta_y^2 u_{ij}^{n+1})+\frac{b}{2h^2}(\delta_x^2 u_{ij}^n+\delta_y^2 u_{ij}^n)$$

这两个格式的截断误差分别为 $O(\tau+h^2)$ 和 $O(\tau^2+h^2)$，并都是绝对稳定的．但是，用差分格式（12.3.25）和（12.3.36）来求解问题（12.3.8），其形成的线性代数方程组的系数矩阵已不是三对角线矩阵，所以不能用追赶法求解．由于在每一时间层上，求解线性代数方程组是费事的，

所以隐式格式在实际计算中也是不方便的.

经常使用的格式，一般都具有绝对稳定、有合理的精度、得到的代数方程组容易求解等优点，交替方向隐式格式就是其中之一。如，逼近问题（12.3.8）中扩散方程的Peaceman-Rachford(佩斯曼-瑞奇福尔德）格式

12.3.27
$$\begin{cases} \dfrac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\tau/2} = \dfrac{b}{h^2}\delta_x^2 u_{ij}^{n+\frac{1}{2}} + \dfrac{b}{h^2}\delta_y^2 u_{ij}^{n} \\ \dfrac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\tau/2} = \dfrac{b}{h^2}\delta_x^2 u_{ij}^{n+\frac{1}{2}} + \dfrac{b}{h^2}\delta_y^2 u_{ij}^{n+1} \end{cases}$$

这个格式的截断误差是$O(\tau^2+h^2)$，绝对稳定.由（12.3.27）的表达式可知，在一个时间步长上，只要用二次追赶法求解就可以解出方程，因而计算容易，类似地但各有特色的格式不少，如表（12.3.28）所示，这类格式一般统称为分裂格式.

12.3.28 表

名称	差分格式与截断误差	稳定条件
Peaceman —Rachford 交替方向隐式 格式	$\dfrac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\tau/2} = \dfrac{b}{h^2}(\delta_x^2 u_{ij}^{n+\frac{1}{2}} + \delta_y^2 u_{ij}^{n})$ $\dfrac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\tau/2} = \dfrac{b}{h^2}(\delta_x^2 u_{ij}^{n+\frac{1}{2}} + \delta_y^2 u_{ij}^{n+1})$ $E=O(\tau^2+h^2)$	绝对稳定
Douglas —Rachford 格式	$\dfrac{u_{ij}^{n+\frac{1}{2}} - u_{ij}^{n}}{\tau} = \dfrac{b}{h^2}[\delta_x^2 u_{ij}^{n+\frac{1}{2}} + \delta_y^2 u_{ij}^{n}]$ $\dfrac{u_{ij}^{n+1} - u_{ij}^{n+\frac{1}{2}}}{\tau} = \dfrac{b}{h^2}[\delta_y^2 u_{ij}^{n+1} - \delta_y^2 u_{ij}^{n}]$ $E=O(\tau+h^2)$	绝对稳定

续表

名称	差分格式与截断误差	稳定条件
局部一维格式	$\frac{u_{ij}^{n+\frac{1}{2}}-u_{ij}^{n}}{\tau}=\frac{b}{h^2}\delta_x^2\left(\frac{u_{ij}^{n+\frac{1}{2}}+u_{ij}^{n}}{2}\right)$ $\frac{u_{ij}^{n+1}-u_{ij}^{n+\frac{1}{2}}}{\tau}=\frac{b}{h^2}\delta_y^2\left(\frac{u_{ij}^{n+1}+u_{ij}^{n+\frac{1}{2}}}{2}\right)$ $E=O(\tau^2+h^2)$	绝对稳定
Douglas 格式	$\frac{u_{ij}^{n+\frac{1}{2}}-u_{ij}^{n}}{\tau}=\frac{b}{2h^2}\delta_x^2(u_{ij}^{n+\frac{1}{2}}+u_{ij}^{n})+\frac{b}{h^2}\delta_y^2 u_{ij}^{n}$ $\frac{u_{ij}^{n+1}-u_{ij}^{n+\frac{1}{2}}}{\tau}=\frac{b}{2h^2}\delta_y^2(u_{ij}^{n+1}-u_{ij}^{n})$ $E=O(\tau^2+h^2)$	绝对稳定
Яненко 格式	$\frac{u_{ij}^{n+\frac{1}{4}}-u_{ij}^{n}}{\tau/2}=\frac{b}{h^2}\delta_x^2 u_{ij}^{n+\frac{1}{4}}$ $\frac{u_{ij}^{n+\frac{1}{2}}-u_{ij}^{n+\frac{1}{4}}}{\tau/2}=\frac{b}{h^2}\delta_y^2 u_{ij}^{n+\frac{1}{2}}$ $\frac{u_{ij}^{n+1}-u_{ij}^{n}}{\tau}=\frac{b}{h^2}(\delta_x^2+\delta_y^2)u_{ij}^{n+\frac{1}{2}}$ $E=O(\tau^2+h^2)$	绝对稳定

12.4

12.4 有限元方法

解椭圆型边值问题的有限元方法，可以看成是变分原理与函数分片多项式插值逼近方法的结合。在几何和物理条件比较复杂的问题中，有限元方法比差分方法有更广泛的适应性。

12.4.1 椭圆型边值问题的变分原理

在xy平面的区域Ω上考虑如下的边值问题

12.4.1 $-\left[\dfrac{\partial}{\partial x}\left(k\dfrac{\partial u}{\partial x}\right)+\dfrac{\partial}{\partial y}\left(k\dfrac{\partial u}{\partial y}\right)\right]=f,\quad (x,y)\in\Omega$

12.4.2 $u=0,\ (x,y)\in\partial\Omega_1$

12.4.3 $\left(k\dfrac{\partial u}{\partial n}+\sigma u\right)=g,\quad (x,y)\in\partial\Omega_2$

这里Ω的边界$\partial\Omega$分为互不重迭的两部分$\partial\Omega_1$和$\partial\Omega_2$，n是$\partial\Omega$上的外法线方向，k是(x,y)的一阶导数连续的函数，且$k(x,y)\geqslant k_0>0$；σ是连续函数，且$\sigma(x,y)\geqslant 0$；f与g均为给定的函数。

对应于边值问题（12.4.1）—（12.4.3），有如下的变分问题：

12.4.4 **Галёркин(伽辽金)变分问题** 求$u\in V$，使得$D(u,v)-F(v)=0$ 对一切$v\in V$成立，其中

$$V=\{v\mid v\in C^1(\Omega),v\mid_{\partial\Omega_1}=0\}$$

$$D(u,v)=\iint_\Omega k\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right)dxdy+\int_{\partial\Omega_2}\sigma uv ds$$

$$F(v)=\iint_\Omega fvdxdy+\int_{\partial\Omega_2}gvds$$

2.4.5 **Ritz(里兹)变分问题** 求$u\in V$，使得$J(u)\leqslant J(v)$对一切$v\in V$成立。其中V仍为（12.4.4）所示，而

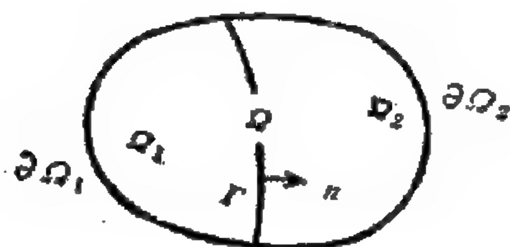
$$J(v)=\frac{1}{2}D(v,v)-F(v)$$

如果把边值问题（12.4.1）—(12.4.3)的物理意义理解为弹性薄膜的平衡问题，则（12.4.4）和（12.4.5）可以分别理解为**虚功原理**和**最小势能原理**。它们以不同的形式反映同一物理现象。

12.4.6 定理 如果$u\in C^2(\Omega)$是边值问题(12.4.1)—(12.4.3)的解，则u是变分问题（12.4.4）的解.反之，如果u是变分问题（12.4.4）的解，且$u\in C^2(\Omega)$，则u是边值问题（12.4.1）—（12.4.3）的解.如果u是变分问题（12.4.4）的解，则u是变分问题(12.4.5)的解，反之亦然.

如果在微分方程（12.4.1）中的系数$k(x, y)$有间断，它的间断线将Ω分为几个子域，为简化问题，设间断线Γ将Ω分为Ω_1和Ω_2，并规定Γ的法线方向，如图（12.4.7）所示.

12.4.7 图



规定在Γ上函数值$(\cdot)^i$为该函数当自变量从Ω_i趋于Γ时的的极限值（$i=1, 2$），列出间断系数的边值问题：

12.4.8

$$-\left[\frac{\partial}{\partial x}\left(k\frac{\partial u}{\partial x}\right)+\frac{\partial}{\partial y}\left(k\frac{\partial u}{\partial y}\right)\right]=f,\quad (x,y)\in\Omega$$

$$u=0,\quad (x,y)\in\partial\Omega_1$$

$$k\frac{\partial u}{\partial n}+\sigma u=g,\quad (x,y)\in\partial\Omega_2$$

$$u^1=u^2,\quad (x,y)\in\Gamma$$

$$\left(k\frac{\partial u}{\partial n}\right)^1=\left(k\frac{\partial u}{\partial n}\right)^2,\quad (x,y)\in\Gamma$$

12.4.9 定理 在系数k间断的情况下，边值问题(12.4.8)对应的变分问题仍为（12.4.4）与（12.4.5）.

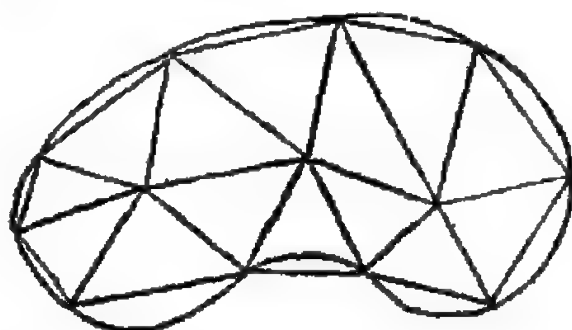
所以，根据变分原理用有限元方法处理间断系数的边值问题，与非间断系数的边值问题的处理方法是同样的，并不带来新的麻烦.

12.4.2 三角形线性元

用有限元方法解二维椭圆型边值问题，常用的、最简单的方法是对区域 Ω 作三角形剖分，在每个三角形单元上作线性插值.

将 Ω 剖分为一组三角形的组合，Ω 的边界就以折线替代，如图（12.4.10）所示.设这组三角形最大边长为h.

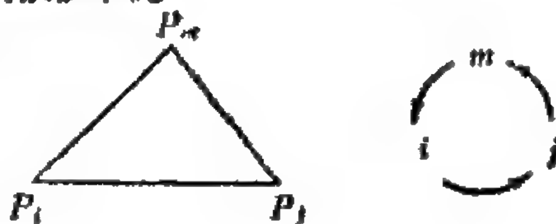
12.4.10 图 Ω 的三角形剖分



每个剖分后的三角形称为一个单元，它的顶点称为节点.对单元和节点进行编号，设有NE个单元e_k（$k=1,2,\cdots,NE$）；有NP个节点P_i，其坐标为（x_i，y_i）（$i=1,2,\cdots,NP$）.剖分时注意不要出现大钝角的三角形，同时可根据物理量的变化布置节点的疏密，在关键部位可加密，其它部位放疏，但是疏密的过渡不要太陡.

用区域 Ω 上的分片线性插值函数近似 u，这样的函数在 Ω 上是连续的，且在每个单元e_k上都是x,y的一次多项式，把这样的分片线性多项式记为$u_h(x,y)$.为了在每个单元 e_k 上列出$u_h(x,y)$的表达式，设 e 是任意一个单元.其顶点为P_i，P_j,P_m，并规定i,j,m按逆时针顺序排列如图（12.4.11）

12.4.11 图 三角形单元



设$u_h(x, y)$在三角形单元e的三顶点上之值为

$$u_i=u_h(x_i,y_i),u_j=u_h(x_j,y_j),u_m=u_h(x_m,y_m)$$

三角形单元e的面积为

$$\Delta_e=\frac{1}{2}\begin{vmatrix} x_i & y_i & 1 \\ x_j & y_j & 1 \\ x_m & y_m & 1 \end{vmatrix}$$

则在单元e上，有

12.4.12 $u_h(x,y)=N_i(x,y)u_i+N_j(x,y)u_j+N_m(x,y)u_m$

其中 $N_i(x,y),N_j(x,y),N_m(x,y)$ 为：

12.4.13 线性插值基函数/Basis Function of Linear Interpolation

$$N_i(x,y)=\frac{1}{2\Delta_e}(a_ix+b_iy+c_i)$$

$$N_j(x,y)=\frac{1}{2\Delta_e}(a_jx+b_jy+c_j)$$

$$N_m(x,y)=\frac{1}{2\Delta_e}(a_mx+b_my+c_m)$$

其中

$$a_i=y_j-y_m,\quad a_j=y_m-y_i,\quad a_m=y_i-y_j$$

$$b_i=x_m-x_j,\quad b_j=x_i-x_m,\quad b_m=x_j-x_i$$

$$c_i=x_jy_m-x_my_j,\quad c_j=x_my_i-x_iy_m,\quad c_m=x_iy_j-x_jy_i$$

12.4.14 线性插值基函数的性质

$$N_k(x_l,y_l)=\begin{cases} 0 & (k\neq l), \\ 1 & (k=l), \end{cases}\qquad k,l=i,j,m$$

$$N_i+N_j+N_m=1$$

$$x_iN_i+x_jN_j+x_mN_m=x$$

$$y_iN_i+y_jN_j+y_mN_m=y$$

考虑边值问题（12.4.1）—（12.4.3）中$k(x, y)=1$的情形.从变分问题（12.4.4）出发，以分片线性插值函数

近似u，可得到近似变分问题：求$u_h \in V_h$，使得

$$D(u_h, v_h) - F(v_h) = 0$$

对一切$v_h \in V_h$成立，其中$D(\cdot, \cdot)$与$F(\cdot)$仍如(12.4.4)所示，V_h则是所有在$\partial\Omega_1$上为零的分片线性函数的集合。为了为了将上述变分问题逐个单元计算，记第n个单元为e_n，其边界为∂e_n。若∂e_n中至少有一条边在Ω的边界$\partial\Omega_2$上，则把这段边界记为γ_n。若∂e_n的三条边都不在$\partial\Omega_2$上，规定γ_n为空集，在其上积分为0，这样，近似变分问题成为

12.4.15 近似变分问题/Approximate Variational Problem

求$u_h \in V_h$，使得

$$\sum_{n=1}^{NE}\left\{\iint_{e_n}\left(\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x}+\frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y}\right)dxdy+\int_{\gamma_n}\sigma u_h v_h ds\right\}$$

$$=\sum_{n=1}^{NE}\left\{\iint_{e_n} f v_h dxdy+\int_{\gamma_n} g v_h ds\right\}$$

对一切$v_h \in V_h$成立。

若$k(x, y)$不是常数，以上问题只须稍作修改。Ritz变分问题（12.4.5）的近似问题可类似地写出。

为了逐元计算（12.4.15）中的积分，设e_n为$\Delta P_i P_j P_m$，顶点上u_h，v_h之值分别为u_i，u_j，u_m和v_i, v_j, v_m，并记

$$\{u\}_{e_n} = [u_i, u_j, u_m]^T, \quad \{v\}_{e_n} = [v_i, v_j, v_m]^T$$

$$[N] = [N_i, N_j, N_m]$$

则在e_n上

$$u_h = [N]\{u\}_{e_n}, \quad v_h = [N]\{v\}_{e_n}$$

$$\begin{pmatrix}\dfrac{\partial u_h}{\partial x}\\[2ex]\dfrac{\partial u_h}{\partial y}\end{pmatrix} = [B]\{u\}_{e_n}, \quad \begin{pmatrix}\dfrac{\partial v_h}{\partial x}\\[2ex]\dfrac{\partial v_h}{\partial y}\end{pmatrix} = [B]\{v\}_{e_n}$$

$$[B]=\frac{1}{2\Delta_e}\begin{pmatrix} a_i & a_j & a_m \\ b_i & b_j & b_m \end{pmatrix}$$

为了方便起见，如γ_n非空集，设γ_n是$\overline{P_iP_j}$边(其它情形以下计算类似)，引入参数t为$\overline{P_iP_j}$上的弧长，对应P_i有$t=0$，对应P_j有$t=l$，这样可得

$$N_i\Big|_{\overline{P_iP_j}}=1-\frac{t}{l},\quad N_j\Big|_{\overline{P_iP_j}}=\frac{t}{l},\quad N_m\Big|_{\overline{P_iP_j}}=0$$

将这些式子代入（12.4.15）等式左端的积分，可得：

$$\iint_{e_n}\left(\frac{\partial u_h}{\partial x}\frac{\partial v_h}{\partial x}+\frac{\partial u_h}{\partial y}\frac{\partial v_h}{\partial y}\right)dxdy+\int_{\gamma_n}\sigma u_h v_h ds$$

$$=\{v\}^T_{e_n}[K]_{e_n}\{u\}_{e_n}$$

其中$[K]_{e_n}$为单元刚度矩阵。

2.4.16 单元刚度矩阵/Element Stiffness Matrix

$$[K]_{e_n}=[\overline{K}]_{e_n}+[\overline{\overline{K}}]_{e_n}$$

其中

$$[\overline{K}]_{e_n}=\begin{pmatrix} \overline{k}_{ii} & \overline{k}_{ij} & \overline{k}_{im} \\ \overline{k}_{ji} & \overline{k}_{jj} & \overline{k}_{jm} \\ \overline{k}_{mi} & \overline{k}_{mj} & \overline{k}_{mm} \end{pmatrix},\quad \overline{k}_{st}=\frac{1}{4\Delta_{e_n}}(a_s a_t+b_s b_t)$$

$$s,t=i,j,m$$

当γ_n为空集时，$[\overline{\overline{K}}]_{e_n}=0$，否则

$$[\overline{\overline{K}}]_{e_n}=\begin{pmatrix} \overline{\overline{k}}_{ii} & \overline{\overline{k}}_{ij} & \overline{\overline{k}}_{im} \\ \overline{\overline{k}}_{ji} & \overline{\overline{k}}_{jj} & \overline{\overline{k}}_{jm} \\ \overline{\overline{k}}_{mi} & \overline{\overline{k}}_{mj} & \overline{\overline{k}}_{mm} \end{pmatrix}$$

$$\overline{\overline{k}}_{ii}=\int_0^l \sigma(1-\frac{t}{l})^2 dt$$

$$\overline{\overline{k}}_{ij}=\overline{\overline{k}}_{ji}=\int_0^l \sigma(1-\frac{t}{l})\frac{t}{l}dt$$

$$\overline{\overline{k}}_{jj}=\int_0^l \sigma\left(\frac{t}{l}\right)^2 dt$$

$$\overline{\overline{k}}_{mi}=\overline{\overline{k}}_{im}=\overline{\overline{k}}_{mj}=\overline{\overline{k}}_{jm}=\overline{\overline{k}}_{mm}=0$$

代入（12.4.15）等式右端的积分可得：

$$\iint_{e_n} f v_h dxdy+\int_{\gamma_n} g v_h ds=\left\{v\right\}_{e_n}^T\left\{F\right\}_{e_n}$$

其中$\{F\}_{e_n}$为单元荷载向量。

12.4.17 单元荷载向量/Element Load Vector

$$\left\{F\right\}_{e_n}=\left\{\overline{F}\right\}_{e_n}+\left\{\overline{\overline{F}}\right\}_{e_n}$$

其中

$$\left\{\overline{F}\right\}_{e_n}=\left[\overline{F}_i^e \quad \overline{F}_j^e \quad \overline{F}_m^e\right]^T, \overline{F}_s^e=\iint_{e_n} N_s f dxdy, s=i,j,m$$

当γ_n为空集时，$\left\{\overline{\overline{F}}\right\}_{e_n}=0$ 否则

$$\left\{\overline{\overline{F}}\right\}_{e_n}=\left[\int_0^l(1-\frac{t}{l})gdt \quad \int_0^l\frac{t}{l}gdt \quad 0\right]^T$$

为了把（12.4.15）中各单元积分迭加起来，令：

$\{u\}=[u_1,\cdots,u_{NP}]^T$，$\{v\}=[v_1,\cdots,v_{NP}]^T$

设单元$e_n=\triangle P_iP_jP_m$，则

$\{v\}_{e_n}=[v_i,v_j,v_m]^T=[C]_{e_n}\{v\}$

其中$[C]_{e_n}$是一个$3\times NP$的矩阵。设 P_i点在节点总编号中序

号为k_l，则$[C]_{e_n}$的第一行第k_l个元素为 1，其余元素 为 0。同理，$[C]_{e_n}$第二、三行只有一个非零元素，其 值 为 1，其位置由P_j，P_m在总编号中的序号决定，同理

$\{u\}_{e_n}=[C]_{e_n}\{u\}$

(12.4.15) 左端积分的迭加得到

$$\sum_{n=1}^{NE}\{v\}^T_{e_n}[K]_{e_n}\{u\}_{e_n}=\{v\}^T[K]\{u\}$$

其中$[K]$称为**刚度矩阵，也称总刚度矩阵。**

12.4.18 **刚度矩阵/Stiffness Matrix**

$$[K]=\sum_{n=1}^{NE}[C]^T_{e_n}[K]_{e_n}[C]_{e_n}$$

(12.4.15) 右端积分的迭加得到

$$\sum_{n=1}^{NE}\{v\}^T_{e_n}\{F\}_{e_n}=\{v\}^T\{F\}$$

其中$[F]$称为荷载向量，也称总荷载向量.

12.4.19 **荷载向量/Load Vector**

$$\{F\}=\sum_{n=1}^{NE}[C]^T_{e_n}\{F\}_{e_n}$$

(12.4.15) 中的等式成为：

12.4.20 $\{v\}^T([K]\{u\}-\{F\})=0$

在实际计算总刚度矩阵$[K]$时，应注意它由$[C]_{e_n}{}^T[K]_{e_n}[C]_{e_n}$迭加而得. $[C]_{e_n}{}^T[K]_{e_n}[C]_{e_n}$是$NP\times NP$的矩 阵，但它只有九个非零元素，这九个元素就是$[K]_{e_n}$的九个元素，它们在$NP\times NP$矩阵中的位置由$P_l$，$P_j$，$P_m$在节点总编号中的序号所确定，所以$[C]_{e_n}{}^T[K]_{e_n}[C]_{e_n}$只是 由 3×3 的矩阵$[K]_{e_n}$"扩大"为$NP\times NP$的矩阵.例如，若计算$e_n$时，$p_l$，$p_j$和$p_m$在节点总编号的序号分别是101，102，95，则有

$$
[C]^T_{e_n}[K]_{e_n}[C]_{e_n}=\begin{bmatrix}
 & \vdots & & \vdots & \vdots & \\
\cdots & k^{e_n}_{mm} & \cdots & k^{e_n}_{mi} & k^{e_n}_{mj} & \cdots \\
 & \vdots & & \vdots & \vdots & \\
\cdots & k^{e_n}_{im} & \cdots & k^{e_n}_{ii} & k^{e_n}_{ij} & \cdots \\
\cdots & k^{e_n}_{jm} & \cdots & k^{e_n}_{ji} & k^{e_n}_{jj} & \cdots \\
 & \vdots & & \vdots & \vdots & \\
 & 95 & & 101 & 102 &
\end{bmatrix}\begin{matrix} \\ 95 \\ \\ 101 \\ 102 \\ \\ \\ \end{matrix}
$$

同理，$[C]^T_{e_n}\{F\}_{e_n}$ 是NP维的向量，它只有三个非零元素，所以它是$\{F\}_{e_n}$的"扩大"。在上面的例子中

$$
[C]^T_{e_n}\{F\}_{e_n}=\begin{bmatrix}
\vdots \\
F^{e_n}_m \\
\vdots \\
F^{e_n}_i \\
F^{e_n}_j \\
\vdots
\end{bmatrix}
$$

为了叙述方便，假设节点编号时，把$\partial\Omega_1$的节点排在最前面，即P_1，…，P_l，其它节点为P_{l+1}，…，P_{NP}。在近似变分问题（12.4.15）中，u_h，v_h都属于V_h，即v_h及u_h在节点P_1，…P_l上的值为0，这样，NP维向量$\{v\}$的前l个分量应为0，即

$\{v\}=[0,\cdots,0,v_{l+1},\cdots,v_{NP}]^T$

$\{u\}$亦类似。这样，（12.4.15）化为如下离散问题。

12.4.21 离散问题/Discrete Problem 求$\{u\}=[0,\cdots,0,u_{l+1},\cdots,u_{NP}]^T$，使得对任意的$\{v\}=[0,\cdots,0,v_{l+1},\cdots,v_{NP}]^T$，$\{v\}^T([K]\{u\}-\{F\})=0$ 成立。

离散问题（12.4.21）可化为求解线性代数方程组的问题。用分块矩阵记号记为

12.4.22 $[K]=\begin{pmatrix} K_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}$, $\{v\}=\begin{pmatrix} v_{\mathrm{I}} \\ v_{\mathrm{II}} \end{pmatrix}$, $\{u\}=\begin{pmatrix} u_{\mathrm{I}} \\ u_{\mathrm{II}} \end{pmatrix}$, $\{F\}=\begin{pmatrix} F_{\mathrm{I}} \\ F_{\mathrm{II}} \end{pmatrix}$

其中K_{11}是$l\times l$的方阵，K_{22}是$(NP-l)\times(NP-l)$的方阵，v_{I},u_{I}和F_{I}都是$l\times1$的，且v_{I}和u_{I}的元素均为0；v_{II},u_{II}和F_{II}均为$(NP-l)\times1$的。因此，离散问题（12.4.21）又可成为求$\{u_{\mathrm{II}}\}=[u_{l+1},\ \cdots,\ u_{NP}]^T$，使得对任意的$\{v_{\mathrm{II}}\}$，

$\{v_{\mathrm{II}}\}^T\{[K_{22}]\{u_{\mathrm{II}}\}-(\{F_{\mathrm{II}}\}-[K_{21}]\{u_{\mathrm{I}}\})\}=0$ 成立。

（12.4.21）具体的求解可以有不同的方法。

12.4.23 离散问题(12.4.21)求解方法之一 求解方程组

$[K_{22}]\{u_{\mathrm{II}}\}=\{F_{\mathrm{II}}\}-[K_{21}]\{u_{\mathrm{I}}\}$

其中系数矩阵$[K_{22}]$是从总刚度矩阵$[K]$中划去头l行l列而得，若$\partial\Omega_1$上的节点不是排在头l个，则在$[K]$中划去相应的l行l列。$\{F_{\mathrm{II}}\}$也类似地从$\{F\}$得到。在齐次边界条件（12.4.8）下，$\{u_{\mathrm{I}}\}$应为0。若边值问题在$\partial\Omega_1$是非齐次约束条件，则$\{u_{\mathrm{I}}\}$是已知的向量。

计算出$[K]$后，再划去l行l列元素，即可得到$[K_{22}]$。这种方法使矩阵的元素存储要重新排列，给编制程序带来麻烦。

12.4.24 离散问题(12.4.21)求解方法之二 求解方程组

$$\begin{pmatrix} I_l & 0 \\ 0 & K_{22} \end{pmatrix}\begin{pmatrix} u_{\mathrm{I}} \\ u_{\mathrm{II}} \end{pmatrix}=\begin{pmatrix} u_{\mathrm{I}}^0 \\ F_{\mathrm{II}}-K_{21}u_{\mathrm{I}}^0 \end{pmatrix}$$

其中I_l是l阶单位矩阵，u_{I}^0由$\partial\Omega_1$上节点u的值所组成。上述方程组系数矩阵保留了$[K]$的阶数，它和方法之一的方程组是等价的。

12.4.25 离散问题(12.4.21)求解方法之三 选择N为一个大数，例如10^{10}，求解方程组

$$\begin{pmatrix} \widetilde{K}_{11} & K_{12} \\ K_{21} & K_{22} \end{pmatrix}\begin{pmatrix} u_{\mathrm{I}} \\ u_{\mathrm{II}} \end{pmatrix}=\begin{pmatrix} \widetilde{F}_{\mathrm{I}} \\ F_{\mathrm{II}} \end{pmatrix}$$

其中分块的方法仍同（12.4.22）.$\widetilde{K}_{11}$是K_{11}的主对角线元素乘N，其它元素不变，即

$$\widetilde{K}_{11}=\begin{pmatrix} k_{11}N & k_{12} & \cdots & k_{1l} \\ & \ddots & & \\ & & \ddots & \\ k_{l1} & k_{l2} & \cdots & k_{ll}\cdot N \end{pmatrix}$$

K_{12}，K_{21}，K_{22}，u_{I}，u_{II}，F_{II}同（12.4.22）而

$$\widetilde{F}_{\mathrm{I}}=\begin{pmatrix} u_1^0\cdot k_{11}\cdot N \\ \vdots \\ u_l^0\cdot k_{ll}\cdot N \end{pmatrix}$$

其中u_1^0，…，u_l^0为u在$\partial\Omega_l$上节点的已知值．解这样的方程组，其实际效果和（12.4.23）相同，且系数矩阵仍保持对称性．

如果$\partial\Omega_l$上的节点不是排在头l个，则（12.4.24）和（12.4.25）的方程组应作适当的行与列的置换．

12.4.26 例 考虑矩形域上无热源的定常温度场，边界上给出绝热条件或给定温度值，求解其温度分布．用线性元方法求解

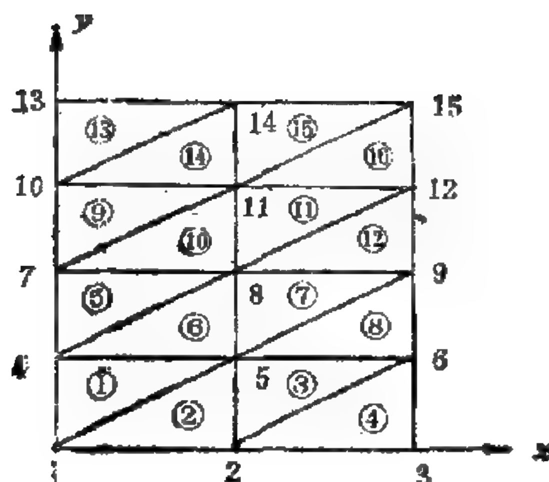
$$\frac{\partial^2 u}{\partial x^2}+\frac{\partial^2 u}{\partial y^2}=0\quad(0<x<2, 0<y<2)$$

$$u=50\ (y=0),\quad u=100(y=2)$$

$$\frac{\partial u}{\partial x}=0\ (x=0),\quad \frac{\partial u}{\partial x}=0\ (x=2)$$

解 对区域$0\leqslant x\leqslant 2, 0\leqslant y\leqslant 2$作三角形剖分，共有十六个三角形单元，十五个节点，单元和节点编号如图（12.4.27）所示．

12.4.27 图 区域的三角形剖分



节点坐标列在表（12.4.28），每个单元的参数a_i，a_j，a_m，b_i，b_j及b_m列成表（12.4.29）。

12.4.28 表 节点坐标

节 点	x	y	节 点	x	y	节 点	x	y
1	0	0	6	2	0.5	11	1	1.5
2	1	0	7	0	1	12	2	1.5
3	2	0	8	1	1	13	0	2
4	0	0.5	9	2	1	14	1	2
5	1	0.5	10	0	1.5	15	2	2

12.4.29 表 单元参数

单元	节点号			参数					
	i	j	m	a_i	a_j	a_m	b_i	b_j	b_m
1	4	1	5	−0.5	0	0.5	1	−1	0
2	2	5	1	0.5	0	−0.5	−1	1	0

单元	节点号			参数					
	i	j	m	a_i	a_j	a_m	b_i	b_j	b_m
3	5	2	6	−0.5	0	0.5	1	−1	0
4	3	6	2	0.5	0	−0.5	−1	1	0
5	7	4	8	−0.5	0	0.5	1	−1	0
6	5	8	4	0.5	0	−0.5	−1	1	0
7	8	5	9	−0.5	0	0.5	1	−1	0
8	6	9	5	0.5	0	−0.5	−1	1	0
9	10	7	11	−0.5	0	0.5	1	−1	0
10	8	11	7	0.5	0	−0.5	−1	1	0
11	11	8	12	−0.5	0	0.5	1	−1	0
12	9	12	8	0.5	0	−0.5	−1	1	0
13	13	10	14	−0.5	0	0.5	1	−1	0
14	11	14	10	0.5	0	−0.5	−1	1	0
15	14	11	15	−0.5	0	0.5	1	−1	0
16	12	15	11	0.5	0	−0.5	−1	1	0

按照（12.4.16）计算单元刚度矩阵时，由于在本例边界条件中$\sigma=g=0$，所以变分问题中不出现γ_e上的线积分项，即$[\overline{\overline{K}}]_{e_n}$为0。在表（12.4.29）的节点排列中，所有奇数单元和偶数单元的参数分别都相同，三角形元的面积亦相同，所以奇数单元的单元刚度矩阵都是一样的，偶数单元也是这样。可算得

$$[K]_{e_1}=[K]_{e_2}=\frac{1}{4}\begin{pmatrix} 5 & -4 & -1 \\ -4 & 4 & 0 \\ -1 & 0 & 1 \end{pmatrix}$$

第1号单元的节点P_i，P_j，P_m为第4，1，5号节点，单元刚度矩阵“扩大”后得

$$
\frac{1}{4}\begin{bmatrix}
4 & \cdot & \cdot & -4 & 0 & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \\
-4 & \cdot & \cdot & 5 & -1 & \\
0 & \cdot & \cdot & -1 & 1 & \\
 & & & & & \ddots
\end{bmatrix}
$$

同理，第2、3号单元的刚度矩阵“扩大”后得

$$
\frac{1}{4}\begin{bmatrix}
1 & -1 & \cdot & \cdot & 0 & \\
-1 & 5 & \cdot & \cdot & -4 & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \\
0 & -4 & \cdot & \cdot & 4 & \\
 & & & & & \ddots
\end{bmatrix},
$$

$$
\frac{1}{4}\begin{bmatrix}
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & 4 & \cdot & \cdot & -4 & 0 & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & \cdot & \cdot & \cdot & \cdot & \cdot & \\
\cdot & -4 & \cdot & \cdot & 5 & -1 & \\
\cdot & 0 & \cdot & \cdot & -1 & 1 & \\
 & & & & & & \ddots
\end{bmatrix}
$$

其它单元刚度矩阵类似地“扩大”，迭加起来得到总刚度矩阵

$$\frac{1}{4}\begin{bmatrix}
5 & -1 & 0 & -4 & & & & & & & & & & & \\
-1 & 10 & -1 & 0 & -8 & & & & & & & & & & \\
0 & -1 & 5 & 0 & 0 & -4 & & & & & & & & & \\
-4 & 0 & 0 & 10 & -2 & 0 & -4 & & & & & & & & \\
 & -8 & 0 & -2 & 20 & -2 & 0 & -8 & & & & & & & \\
 & & -4 & 0 & -2 & 10 & 0 & 0 & -4 & & & & & & \\
 & & & -4 & 0 & 0 & 10 & -2 & 0 & -4 & & & & & \\
 & & & & -8 & 0 & -2 & 20 & -2 & 0 & -8 & & & & \\
 & & & & & -4 & 0 & -2 & 10 & 0 & 0 & -4 & & & \\
 & & & & & & -4 & 0 & 0 & 10 & -2 & 0 & -4 & & \\
 & & & & & & & -8 & 0 & -2 & 20 & -2 & 0 & -8 & \\
 & & & & & & & & -4 & 0 & -2 & 10 & 0 & 0 & -4 \\
 & & & & & & & & & -4 & 0 & 0 & 5 & -1 & 0 \\
 & & & & & & & & & & -8 & 0 & -1 & 10 & -1 \\
 & & & & & & & & & & & -4 & 0 & -1 & 5
\end{bmatrix}$$

进行约束条件处理时，注意边界点是1，2，3和13，14，15六个点，用方法之二（12.4.24），有 $u_1^0=u_2^0=u_3^0=50$，$u_{13}^0=u_{14}^0=u_{15}^0=100$，使总刚度矩阵中第1，2，3，13，14，15行和到对角线元素改为1，其它元素 改 为 0；右 端 亦 按（12.4.24）计算，得到方程组

$$
\frac{1}{4}\begin{pmatrix}
4 & & & & & & & & & & & & & & \\
 & 4 & & & & & & & & & & & & & \\
 & & 4 & & & & & & & & & & & & \\
 & & & 10 & -2 & 0 & -4 & & & & & & & & \\
 & & & -2 & 20 & -2 & 0 & -8 & & & & & & & \\
 & & & 0 & -2 & 10 & 0 & 0 & -4 & & & & & & \\
 & & & -4 & 0 & 0 & 10 & -2 & 0 & -4 & & & & & \\
 & & & & -8 & 0 & -2 & 20 & -2 & 0 & -8 & & & & \\
 & & & & & -4 & 0 & -2 & 10 & 0 & 0 & -4 & & & \\
 & & & & & & -4 & 0 & 0 & 10 & -2 & 0 & & & \\
 & & & & & & & -8 & 0 & -2 & 20 & -2 & & & \\
 & & & & & & & & -4 & 0 & -2 & 10 & & & \\
 & & & & & & & & & & & & 4 & & \\
 & & & & & & & & & & & & & 4 & \\
 & & & & & & & & & & & & & & 4
\end{pmatrix}
\begin{pmatrix} u_1 \\ u_2 \\ u_3 \\ u_4 \\ u_5 \\ u_6 \\ u_7 \\ u_8 \\ u_9 \\ u_{10} \\ u_{11} \\ u_{12} \\ u_{13} \\ u_{14} \\ u_{15} \end{pmatrix}
=
\begin{pmatrix} 50 \\ 50 \\ 50 \\ 50 \\ 100 \\ 50 \\ 0 \\ 0 \\ 0 \\ 100 \\ 200 \\ 100 \\ 100 \\ 100 \\ 100 \end{pmatrix}
$$

这个方程组的解是

$u_1=u_2=u_3=50$，$u_4=u_5=u_6=62.5$

$u_7=u_8=u_9=75$，$u_{10}=u_{11}=u_{12}=87.5$

$u_{13}=u_{14}=u_{15}=100$

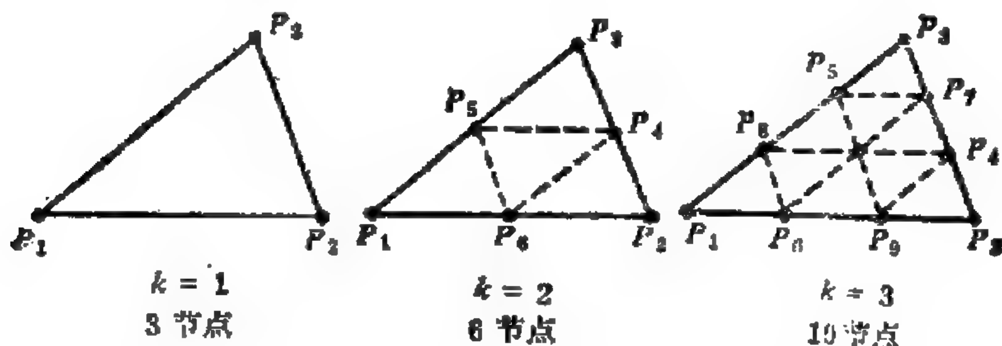
12.4.3 三角形单元上的高次插值

三角形单元除了用上述线性多项式插值外，还可用 x,y 的 k 次完全多项式插值．k 次完全多项式有 $\frac{1}{2}(k+1)(k+2)$ 项，即

$$
\begin{array}{ccccccccc}
 & & & 1 & & & & & 0\text{次} \\
 & & x & & y & & & & 1\text{次} \\
 & x^2 & & xy & & y^2 & & & 2\text{次} \\
x^3 & & x^2y & & xy^2 & & y^3 & & 3\text{次} \\
\cdots & & \cdots & & \cdots & & \cdots & \cdots &
\end{array}
$$

故 k 次Lagrange插值，需要 $\frac{1}{2}(k+1)(k+2)$ 个节点．如图(12.4.30）所示．

12.4.30 图 三角形Lagrange插值的节点



设三角形元 $e=\Delta P_1P_2P_3$，其面积为 Δe，节点 P_i 坐标为 (x_i,y_i)，e 内一点 $P(x,y)$．

12.4.31 定义/面积坐标/Area Coordinate e内一点$P(x,y)$的面积坐标 (L_1,L_2,L_3) 为：

$$L_1=\frac{1}{2\Delta_e}\begin{vmatrix} x & y & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{vmatrix}, \quad L_2=\frac{1}{2\Delta_e}\begin{vmatrix} x & y & 1 \\ x_3 & y_3 & 1 \\ x_1 & y_1 & 1 \end{vmatrix}$$

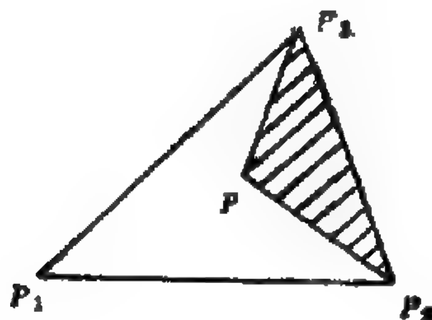
$$L_3=\frac{1}{2\Delta_e}\begin{vmatrix} x & y & 1 \\ x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \end{vmatrix}$$

L_1,L_2,L_3即为(12.4.13) 的N_i,N_j,N_m，所以由(12.4.14)，

$$L_j(P_i)=\begin{cases}0 & (i\neq j) \\ 1 & (i=j)\end{cases} \qquad (i,j=1,2,3)$$

$$\sum_{i=1}^{3}L_i=1, \quad \sum_{i=1}^{3}L_ix_i=x, \quad \sum_{i=1}^{3}L_iy_i=y$$

12.4.32　图　面积坐标的几何意义



$$L_1=\frac{\Delta PP_2P_3\text{的面积}}{\Delta P_1P_2P_3\text{的面积}}$$

$$L_2=\frac{\Delta PP_1P_3\text{的面积}}{\Delta P_1P_2P_3\text{的面积}}$$

$$L_3=\frac{\Delta PP_1P_2\text{的面积}}{\Delta P_1P_2P_3\text{的面积}}$$

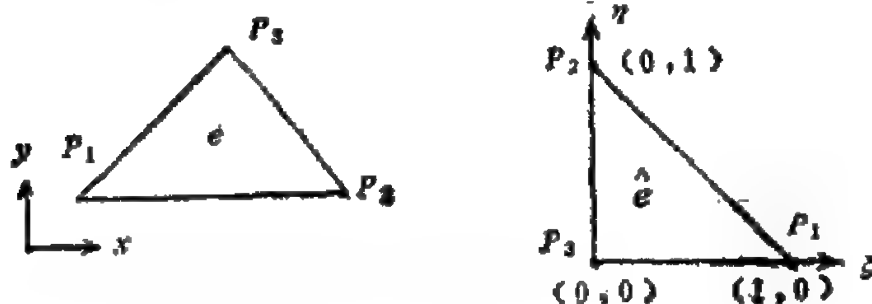
12.4.33　$\begin{cases}\xi=L_1(x,y) \\ \eta=L_2(x,y)\end{cases}$

是 (x,y) 平面到 (ξ,η) 平面的变换，其逆变换为：

$$\begin{cases} x=(x_1-x_3)\xi+(x_2-x_3)\eta+x_3 \\ y=(y_1-y_3)\xi+(y_2-y_3)\eta+y_3 \end{cases}$$

(12.4.33) 将 (x,y) 平面上三角形元 e 变换到 (ξ,η) 平面上标准三角形 $\hat{e}$，如图 (12.4.34) 所示.

12.4.34 图



在近似变分问题中，积分的计算可以在 $\hat{e}$ 上进行，积分公式为

12.4.35 $$\iint_e F(L_1(x,y),L_2(x,y)L(x,y))dxdy$$

$$=2\Delta_e\int_0^1 d\xi\int_0^{1-\xi}F(\xi,\eta,1-\xi-\eta)d\eta$$

其中一个重要例子是

12.4.36 $$\iint_e L_1^{\lambda_1}L_2^{\lambda_2}L_3^{\lambda_3}dxdy=\frac{\lambda_1!\lambda_2!\lambda_3!}{(\lambda_1+\lambda_2+\lambda_3+2)!}\cdot 2\Delta_e$$

式中 $\lambda_1,\lambda_2,\lambda_3$ 为非负整数.

对三角形单元，经常用到数值积分方法，表(12.4.37)列举了一些数值积分公式.

12.4.37 表 三角形上的数值积分公式

$$\iint_e F(L_1,L_2,L_3)dxdy=\Delta_e\sum_{k=1}^{m}\rho^{(k)}F(L_1^{(k)},L_2^{(k)},L_3^{(k)})$$

节点个数 m	节点坐标 (L_1, L_2, L_3)	权数 ρ	精度次数 n
1	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	1	1
3	$\left(\frac{1}{2}, \frac{1}{2}, 0\right)$	$\frac{1}{3}$	2
	$\left(0, \frac{1}{2}, \frac{1}{2}\right)$	$\frac{1}{3}$	
	$\left(\frac{1}{2}, 0, \frac{1}{2}\right)$	$\frac{1}{3}$	
4	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	$-\frac{27}{48}$	3
	(0.6, 0.2, 0.2) (0.2, 0.6, 0.2) (0.2, 0.2, 0.6)	$\frac{25}{48}$	
7	$\left(\frac{1}{3}, \frac{1}{3}, \frac{1}{3}\right)$	0.2250000000	5
$\alpha_1=0.0597158717$	$(\alpha_1, \beta_1, \beta_1)$ $(\beta_1, \alpha_1, \beta_1)$ $(\beta_1, \beta_1, \alpha_1)$	0.1323941527	
$\beta_1=0.4701420641$ $\alpha_2=0.7974269853$ $\beta_2=0.1012865073$	$(\alpha_2, \beta_2, \beta_2)$ $(\beta_2, \alpha_2, \beta_2)$ $(\beta_2, \beta_2, \alpha_2)$	0.1259391805	

表中的精度次数 n 指公式对 n 次多项式是准确的。

在三角形单元 e 上，如果是（12.4.30）中 $k=2$ 的情形，则有六个节点，节点 P_i 的面积坐标分别为 $(1,0,0)$，$(0,1,0)$，$(0,0,1)$，$\left(0,\frac{1}{2},\frac{1}{2}\right)$，$\left(\frac{1}{2},0,\frac{1}{2}\right)$，$\left(\frac{1}{2},\frac{1}{2},0\right)$。

若 u 在节点 P_i 上之值为 u_i $(i=1,\cdots,6)$，则在 e 上 u 的二次插值函数为

12.4.38 $u_h=\sum_{i=1}^{6} u_i N_i$

其中 N_i 为三角形单元上的二次插值基函数。

12.4.39 二次插值基函数/Basis Function of Quadrtic Interpolation

$N_i=L_i(2L_i-1) \quad (i=1,2,3)$

$N_4=4L_2L_3, \quad N_5=4L_3L_1, \quad N_6=4L_1L_2$

由（12.4.38）式，可以类似线性插值情形，得到近似变分问题和对应的离散问题。

如果是图（12.4.30）$k=3$情形，则10个节点面积坐标分别为(1,0,0)，(0,1,0)，(0,0,1)，$\left(0,\frac{2}{3},\frac{1}{3}\right)$，$\left(\frac{1}{3},0,\frac{2}{3}\right)$，$\left(\frac{2}{3},\frac{1}{3},0\right)$，$\left(0,\frac{1}{3},\frac{2}{3}\right)$，$\left(\frac{2}{3},0,\frac{1}{3}\right)$，$\left(\frac{1}{3},\frac{2}{3},0\right)$，$\left(\frac{1}{3},\frac{1}{3},\frac{1}{3}\right)$。

12.4.40 三次插值基函数/Basis Functions of Cubic Interpolation

$N_i=\frac{1}{2}L_i(3L_i-1)(3L_i-2) \quad (i=1,2,3)$

$N_4=\frac{9}{2}L_2L_3(3L_2-1)$

$N_5=\frac{9}{2}L_3L_1(3L_3-1)$

$N_6=\frac{9}{2}L_1L_2(3L_1-1)$

$N_7=\frac{9}{2}L_2L_3(3L_3-1)$

$N_8=\frac{9}{2}L_3L_1(3L_1-1)$

$$N_9=\frac{9}{2}L_1L_2(3L_2-1)$$

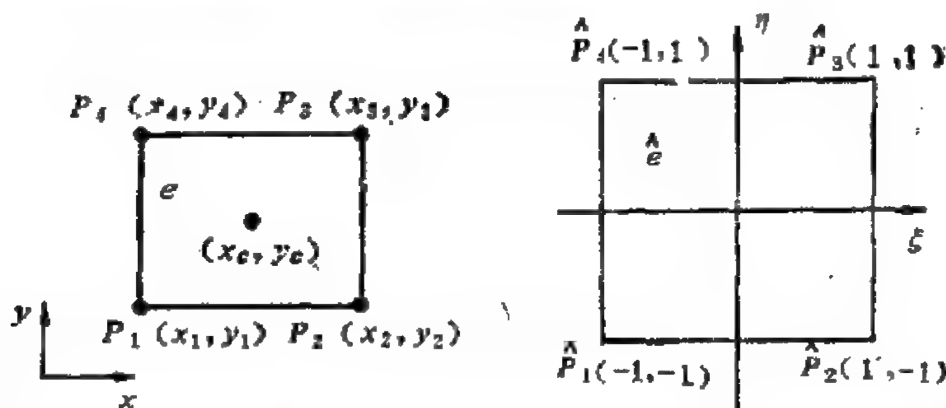
$$N_{10}=27L_1L_2L_3$$

在 e 上，u 的三次插值函数表示为$u_h=\sum_{i=1}^{10}u_iN_i$.在作了三角形剖分的 Ω 上，得到分片三次插值函数 u_h.在一个单元的一条边，它可表示为弧长参数 t 的三次函数，由边上四个节点的 u_i 唯一确定，所以有$u_h\in C(\overline{\Omega})$.分片线性和二次插值也如此。

12.4.4 矩形单元

矩形剖分也是一种应用广泛的剖分方法. 把边值问题(12.4.1)—(12.4.3)的求解区域Ω剖分为若干个矩形单元的组合,这些矩形单元的边都平行于坐标轴,如(12.4.41)所示。其中(x_c,y_c) 是矩形的形心。

12.4.41 图 矩型单元e和标准矩阵单元$\hat{e}$



12.4.42 定义/局部坐标/Local Coordinate 矩形单元 e 中点 $P(x,y)$的局部坐标为

$$\xi=\frac{2(x-x_c)}{x_2-x_1},\quad \eta=\frac{2(y-y_c)}{y_2-y_1}$$

上式可以看成 (x,y) 平面至 (ξ,η) 平面的变换，它将单元 e 变换到标准单元 $\hat{e}$，如图 (12.4.41) 所示。

双线性插值函数共有四项，即在e上$u_h(x,y)=a+bx+cy+dxy$. $1,x,y,xy$四项可看成

$$\begin{pmatrix}1\\x\end{pmatrix}[1\ \ y]=\begin{pmatrix}1 & y\\x & xy\end{pmatrix}$$

当x,y中固定一个时，u_h就是另一个变量的线性函数。

在标准单元$\hat{e}$中，节点$\hat{P}_i$坐标为(ξ_i,η_i) $i=1,2,3,4$，如图（12.4.41）所示，设在$\hat{P}_i$点u的值为u_i，则在$\hat{e}$上有

$$u_h=\sum_{i=1}^{4}N_iu_i$$

其中$N_i(i=1,2,3,4)$为双线性插值基函数。

12.4.43 双线性插值基函数/Basis Functions of Bilinear Interpolation

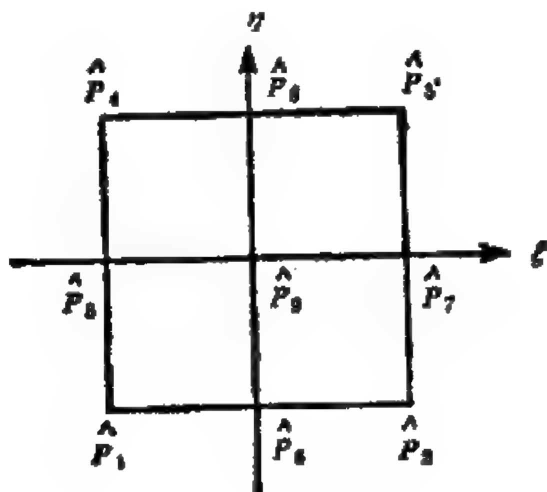
$$N_i=\frac{1}{4}(1+\xi_i\xi)(1+\eta_i\eta)\quad(i=1,2,3,4)$$

矩形单元上双二次插值共有九项，可看成

$$\begin{pmatrix}1\\x\\x^2\end{pmatrix}[1\ \ y\ \ y^2]=\begin{pmatrix}1 & y & y^2\\x & xy & xy^2\\x^2 & x^2y & x^2y^2\end{pmatrix}$$

共需九个节点，如图（12.4.44）所示。

12.4.44 图



12.4.45 双二次插值基函数/Basic Functions of Biquadratic Interpolation

$$N_1 = \frac{1}{4}\xi\eta(\xi-1)(\eta-1)$$

$$N_2 = \frac{1}{4}\xi\eta(\xi+1)(\eta-1)$$

$$N_3 = \frac{1}{4}\xi\eta(\xi+1)(\eta+1)$$

$$N_4 = \frac{1}{4}\xi\eta(\xi-1)(\eta+1)$$

$$N_5 = \frac{1}{2}\xi(\xi-1)(1-\eta^2)$$

$$N_6 = \frac{1}{2}\eta(\eta-1)(1-\xi^2)$$

$$N_7 = \frac{1}{2}\xi(\xi+1)(1-\eta^2)$$

$$N_8 = \frac{1}{2}\eta(\eta+1)(1-\xi^2)$$

$$N_9 = (1-\xi^2)(1-\eta^2)$$

在双二次插值式中去掉 x^2y^2 一项，对应在 $\hat{e}$ 中去掉内部节点 $\hat{P}_9$，得到不完全的双二次插值，它的基函数是

12.4.46 不完全双二次插值的基函数

$$N_1 = -\frac{1}{4}(1-\xi)(1-\eta)(1+\xi+\eta)$$

$$N_2 = -\frac{1}{4}(1+\xi)(1-\eta)(1-\xi+\eta)$$

$$N_3 = -\frac{1}{4}(1+\xi)(1+\eta)(1-\xi-\eta)$$

$$N_4 = -\frac{1}{4}(1-\xi)(1+\eta)(1+\xi-\eta)$$

$$N_5 = \frac{1}{2}(1-\eta^2)(1-\xi)$$

$$N_6 = \frac{1}{2}(1-\xi^2)(1-\eta)$$

$$N_7 = \frac{1}{2}(1-\eta^2)(1+\xi)$$

$$N_8 = \frac{1}{2}(1-\xi^2)(1+\eta)$$

在Ω上作分片线性、二次和不完全二次插值所得的分片插值函数，都属于$C(\overline{\Omega})$。

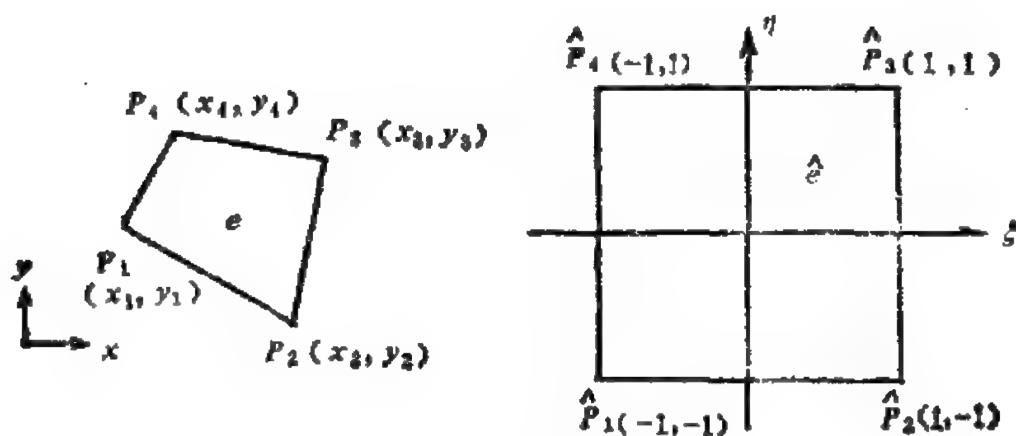
12.4.5 等参数单元

设N_i ($i=1,2,3,4$) 是 (12.4.48) 所示双线性插值基函数，作变换

12.4.47 $$x=\sum_{i=1}^{4} x_i N_i(\xi,\eta), \qquad y=\sum_{i=1}^{4} y_i N_i(\xi,\eta)$$

它将 (x,y) 平面的任意四边形单元e变换到 (ξ,η) 平面的的标准正方形$\hat{e}$，如图 (12.4.48) 所示。

12.4.48 图　四节点四边形的变换



在$\hat{e}$上取插值公式为

12.4.49 $u_h=\sum_{i=1}^{4}u_iN_i(\xi,\eta)$

即可作近似变分问题的计算。这种任意四边形单元的方法称为四节点四边形等参数单元方法。

设N_i $(i=1,2,\cdots,8)$为(12.4.46)的不完全双二次插值基函数，则八节点四边形等参数单元的插值公式为

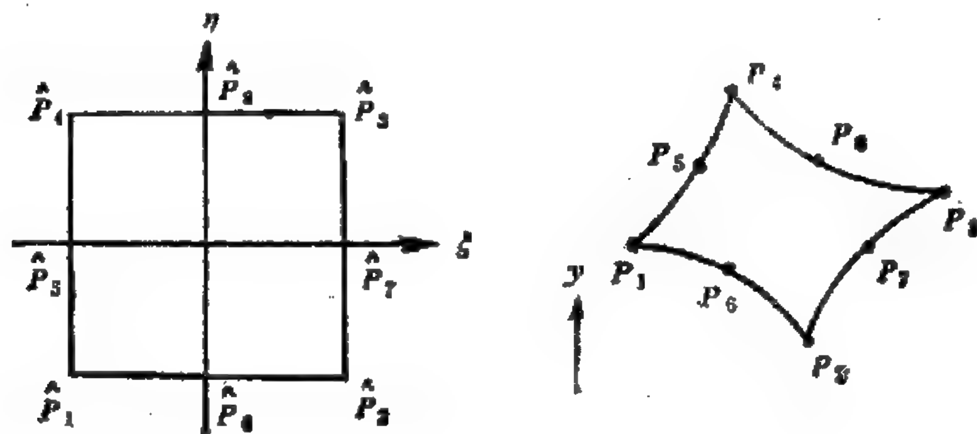
12.4.50 $u_h=\sum_{i=1}^{8}u_iN_i(\xi,\eta)$

对应的变换为

12.4.51 $x=\sum_{i=1}^{8}x_iN_i(\xi,\eta)$，$y=\sum_{i=1}^{8}y_iN_i(\xi,\eta)$

它将图(12.4.52)的e变换至标准单元$\hat{e}$，其中e的边都是二次曲线。

12.4.52 图 八节点四边形的变换



13

第13章　积分方程数值解法

13.1

13.1 引　言

积分方程（Integral Equation）

13.1.1 $y(x)=\lambda\int_a^b K(x,s)y(s)ds+f(x)$

称为**第二类**Fredholm（**弗雷德霍姆**）**积分方程**（Fredholm Integral Equation of the Second Kind），其中y为未知函数，K为**积分方程的核**（Kernel of the Integral Equation），f为右端项或称自由项，λ为参数。λ、K和f均为已知。积分方程（13.1.1）的数值解法即是求未知函数y的近似$\widetilde{y}$。

积分方程

13.1.2 $\int_a^b K(x,s)y(s)ds=\kappa y(x)$

有非零解 y，此时κ为**核的特征值**（Eigenvalue），而y为对应于特征值κ的**特征函数**（Eigenfunction）。方程（13.1.2）的数值解法即是求κ和y的近似解。

积分方程

13.1.3 $y(x)=\lambda\int_0^x K(x,s)y(s)ds+f(x)$

称为**第二类**Volterra（**沃尔泰拉**）**积分方程**（Volterra Integral Equation of Second Kind）。其数值解法即为求近似解$\widetilde{y}$。

13.2

13.2 数值求积法

13.2.1 方法的一般描述

求解第二类Fredholm积分方程

$$y(x)=\lambda\int_a^b K(x,s)y(s)ds+f(x)$$

的**数值求积方法**（Quadrature Method）或称**有限和方法**（Method of Finite Sums）是：取某一数值求积公式

13.2.1 $$\int_a^b F(x)dx\approx\sum_{k=1}^{n}A_kF(x_k)$$

其中$x_k\in[a,b]$是求积的节点，A_k是不依赖于F的求积公式的系数。

把第二类Fredholm积分方程中的定积分用数值求积公式来代替，于是积分方程变为一个近似式

13.2.2 $$y(x)\approx\lambda\sum_{k=1}^{n}A_kK(x,x_k)y(x_k)+f(x)$$

如果把上式写为

13.2.3 $$\tilde{y}(x)=\lambda\sum_{k=1}^{n}A_kK(x,x_k)\tilde{y}(x_k)+f(x)$$

则$\tilde{y}$为y的近似解．令$x=x_1,x_2,\cdots,x_n$，并采用记号
$\tilde{y}_k=\tilde{y}(x_k)$，$K_{ik}=K(x_i,x_k)$，$f_k=f(x_k)$

由（13.2.3）得线性代数方程组

13.2.4 $$\tilde{y}_i=\lambda\sum_{k=1}^{n}A_kK_{ik}\tilde{y}_k+f_i\qquad(i=1,2,\cdots,n)$$

如果方程组（13.2.4）的行列式

13.2.5 $$|I-\lambda L|\neq 0$$

那么方程组有唯一解$\tilde{y}_1,\tilde{y}_2,\cdots,\tilde{y}_n$，其中

$$L=\begin{pmatrix} A_1K_{11} & A_2K_{12} & \cdots & A_nK_{1n} \\ A_1K_{21} & A_2K_{22} & \cdots & A_nK_{2n} \\ \cdots & \cdots & \cdots & \\ A_1K_{n1} & A_2K_{n2} & \cdots & A_nK_{nn} \end{pmatrix}$$

$\widetilde{y}_1,\widetilde{y}_2,\cdots,\widetilde{y}_n$就是积分方程(13.1.1)的解$y$在求积公式的节点$x_1,x_2,\cdots,x_n$上的近似值.利用这些值可以求出任意$x\in[a,b]$的积分方程(13.1.1)的近似值

13.2.6 $$\widetilde{y}(x)=f(x)+\lambda\sum_{k=1}^{n}A_kK(x,x_k)\widetilde{y}_k$$

13.2.7 **例** 用Simpson求积公式和三点Gauss-Legendre求积公式解积分方程

$$y(x)=\int_0^1(1-3xs)y(s)ds+(1-3x)$$

解 Simpson 求积公式为

$$\int_0^1F(x)dx\approx\frac{1}{6}[F(0)+4F(\frac{1}{2})+F(1)]$$

应用这个公式到积分方程得线性代数方程组

$$\begin{cases}\widetilde{y}_1=\frac{1}{6}[\widetilde{y}_1+4\widetilde{y}_2+\widetilde{y}_3]+1\\ \widetilde{y}_2=\frac{1}{6}[\widetilde{y}_1+\widetilde{y}_2-\frac{1}{2}\widetilde{y}_3]-\frac{1}{2}\\ \widetilde{y}_3=\frac{1}{6}[\widetilde{y}_1-2\widetilde{y}_2-2\widetilde{y}_3]-2\end{cases}$$

解这个方程组，得$\widetilde{y}_1=\frac{2}{3}$，$\widetilde{y}_2=-\frac{1}{3}$，$\widetilde{y}_3=-\frac{4}{3}$.

利用近似解表达式（13.2.6）可以得出

$$\widetilde{y}(\frac{1}{4})=\frac{1}{6}\ ,\quad \widetilde{y}(\frac{3}{4})=-\frac{5}{6}\ .$$

三点 Gauss-Legendre 求积公式为

$$\int_0^1 F(x)dx \approx \frac{1}{18}[5F(\alpha)+8F(\frac{1}{2})+5F(\beta)]$$

其中

$$\alpha = \frac{1}{2}(1-\sqrt{0.6})=0.112702$$

$$\beta = \frac{1}{2}(1+\sqrt{0.6})=0.887298$$

应用此积分公式到积分方程得线性代数方程组

$$\begin{cases} 0.732807\widetilde{y_1}-0.369310\widetilde{y_2}-0.194444\widetilde{y_3}=0.661895 \\ -0.230819\widetilde{y_1}+0.888889\widetilde{y_2}+0.091930\widetilde{y_3}=-0.5 \\ -0.194444\widetilde{y_1}+0.147088\widetilde{y_2}+1.378304\widetilde{y_3}=-1.661894 \end{cases}$$

解之，得

$\widetilde{y_1}=0.441264$，$\widetilde{y_2}=-0.333333$，$\widetilde{y_3}=-1.107929$

利用近似解表达式可得

$\widetilde{y}(0)=0.666667$，$\widetilde{y}(\frac{1}{4})=0.166667$

$\widetilde{y}(\frac{3}{4})=-0.833334$，$\widetilde{y}(1)=-1.333333$

本例的解析解为$y(x)=\frac{2}{3}(1-3x)$，可以看出求解是很精确的。

由于在求积分方程的近似解的过程中都要解线性代数方程组，而方程的个数与数值求积公式的节点数相同，因此不宜靠增加节点数来提高求积公式的精度，故采用 Gauss 求积公式是较为适合的。

13.2.2 乘积积分法

在积分方程（13.1.1）中要计算积分

$$\int_a^b K(x,s)y(s)ds$$

如果固定 x，令

$$W(s)=K(x,s)\ ,\quad s\in[a,b]$$

那么上述积分就化为通常的乘积积分

$$\int_a^b W(s)y(s)ds$$

由此可以采用乘积积分的方法来计算。

在区间$[a,b]$上取n个不同的节点（通常取等距节点），

$s_1,s_2,\cdots,s_n$

构造求积公式

$$\int_a^b K(x,s)\varphi(s)ds\approx\sum_{k=1}^n A_k(x)\varphi_k$$

其中系数依赖于 x。选取$A_k(x)$使求积公式的代数精度为 n，即要求对$\varphi(s)=s^l$（$l=0,1,\cdots,n-1$）有

13.2.8 $$\sum_{k=1}^n A_k(x)s_k^l=\int_a^b K(x,s)s^l ds\quad (l=0,1,\cdots,n-1)$$

这是一个线性代数方程组，其中

$A_1(x)$，$A_2(x)$，$\cdots,A_n(x)$

是未知函数。当$A_k(x)(k=1,2,\cdots,n)$求出之后，可以用近似式

$$y(x)\approx\lambda\sum_{k=1}^n A_k(x)y_k+f(x)$$

来代替积分方程（13.1.1）。

如果令 $x=s_1,s_2,\cdots,s_n$，则得线性代数方程组

13.2.9 $$\widetilde{y_i}=\lambda\sum_{k=1}^n A_k(s_i)\widetilde{y_k}+f(s_i)\quad (i=1,2,\cdots,n)$$

由此可以求出解的近似值$\widetilde{y_1},\widetilde{y_2},\cdots,\widetilde{y_n}$。

为了计算$A_k(s_i)$ $(k,i=1,2,\cdots,n)$，一般要求核$K(x,s)$有解析表达式。

13.2.10 **例** 用乘积积分法解积分方程

$$y(x)=-\frac{1}{\pi}\int_{-1}^{1}\frac{y(s)}{1+(x-s)^2}ds+1$$

解 采用十一个等距节点，得出十一个未知数的十一个方程所构成的方程组，解之得

$\tilde{y}(0)=0.65741$，　　$\tilde{y}(\pm 0.2)=0.66151$

$\tilde{y}(\pm 0.4)=0.67389$，$\tilde{y}(\pm 0.6)=0.69448$

$\tilde{y}(\pm 0.8)=0.72249$，$\tilde{y}(\pm 1)=0.75572$

这个结果是相当精确的。

13.2.3 修正的数值求积方法

使用数值求积方法时，已假定核函数K是充分光滑的。如果K有奇异性质，则要进行一些辅助工作以抵消奇异性质的影响。而乘积积分方法可以用于求解具有奇异性质的核K的积分方程。

如果当$x=s$时，K具有奇异性质，则先把积分方程（13.1.1）改写为

13.2.11 $$y(x)-\lambda y(x)\int_a^x K(x,s)ds$$

$$=\lambda\int_a^x K(x,s)[y(s)-y(x)]\,ds+f(x)$$

等号左边的积分是不含未知函数的，因而可以计算出来。而等号右边的积分，由于当$x=s$时有$[y(s)-y(x)]=0$，因此可以消除或减弱奇异性质。

利用数值求积公式（13.2.1），并令$\tilde{y}$为积分方程的近

似解，则可写出

$$\tilde{y}(x)[1-\lambda B(x)]=\lambda\sum_{k=1}^{n}A_kK(x,s_k)[\tilde{y}(s_k)-\tilde{y}(x)]+f(x)$$

其中

$$B(x)=\int_a^b K(x,s)ds$$

如果记

$$\Delta(x)=\sum_{k=1}^{n}A_kK(x,s_k)-B(x)$$

并令 $x=s_j$，那么就得到

13.2.12 $$\tilde{y}(s_j)[1+\lambda\Delta(s_j)]=\lambda\sum_{k=1}^{n}A_kK(s_j,s_k)\tilde{y}(s_k)+f(s_j)$$

$$(j=1,2,\cdots,n)$$

公式（13.2.12）称为**修正的数值求积方法**（Modified Quadrature Method）。

13.2.13 **例** 求积分方程

$$y(x)=-\int_0^1 K(x,s)y(s)ds+x^2$$

的近似解，其中

$$K(x,s)=\begin{cases}x(1-s), & \text{当}0\leqslant x\leqslant s\leqslant 1\\ s(1-x), & \text{当}0\leqslant s\leqslant x\leqslant 1\end{cases}$$

积分方程的精确解为 $y(x)=2\cosh x-0.07335\sinh x-2$。

解 先用一般数值求积方法来求解。利用二个节点的Gauss-Legendre求积公式，节点为 $s_1=0.21132$，$s_2=0.78868$。相应的求积系数为 $A_1=A_2=\frac{1}{2}$，那么可以求得线性代数方程组

$$\begin{cases}1.08333\tilde{y}_1+0.02233\tilde{y}_2=0.04466\\ 0.02233\tilde{y}_1+1.08333\tilde{y}_2=0.62202\end{cases}$$

解此方程组，得其解

$\widetilde{y}_1=0.02940$ ， $\widetilde{y}_2=0.57357$

这两个解与积分方程的解在相应的节点上的值相比较有

$y(s_1)-\widetilde{y}_1=-0.00020$ ， $y(s_2)-\widetilde{y}_2=0.01732$

由于此例中积分方程的核K在 $x=s$ 处一阶导数不连续，为此应采用修正的数值求积方法，把积分方程写为(13.2.11)的形式（$\lambda=-1$）

$$y(x)[1+\int_0^1 K(x,s)ds]=-\int_0^1 K(x,s)[y(s)-y(x)]ds+x^2$$

由于

$$\int_0^1 K(x,s)ds=\frac{x}{2}(1-x)$$

所以上面的积分方程可以写为

$$y(x)\left[1+\frac{x}{2}(1-x)\right]+\int_0^1 K(x,s)[y(x)-y(s)]ds=x^2$$

仍利用二点Gauss-Legendre求积公式代替上述积分，并令$x=s_1,s_2$，这样就得到线性方程组

$$\begin{cases} 1.06100\widetilde{y}_1+0.02233\widetilde{y}_2=0.04466 \\ 0.02233\widetilde{y}_1+1.06100\widetilde{y}_2=0.62202 \end{cases}$$

解这个方程组，得

$\widetilde{y}_1=0.02976$ ， $\widetilde{y}_2=0.58564$

再用积分方程的精确解$y(x)$在求积节点上的值与其比较，可以得

$y(s_1)-\widetilde{y}_1=-0.00056$ ， $y(s_2)-\widetilde{y}_2=0.00525$

由此可以看出，采用修正的数值求积方法得到的数值结果更精确。

13.2.4 重迭核方法

当积分方程的核K有奇点，而又能够算出**重迭核**（Iterated Kernel）

$$K_2(x,s)=\int_a^b K(x,t)K(t,s)dt$$

时，那么可以把积分方程（13.1.1）化为如下形式

13.2.14 $$y(x)=\lambda^2\int_a^b K_2(x,s)y(s)ds+\lambda\int_a^b K(x,s)f(s)ds+f(x)$$

其中积分

$$\int_a^b K(x,s)f(s)dx$$

不包含未知函数，因而可以计算出来。而含重迭核K_2的积分用数值求积会比含K的积分更有效，这个方法称为**重迭核方法**（Iterated Kernel Method）。

13.2.15 **例** 用重迭核方法求例（13.2.13）中积分方程的近似解。

解 积分方程的重迭核为

$$K_2(x,s)=\begin{cases}\dfrac{1}{6}(xs^3+sx^3-3xs^2+2xs-x^3), & 当0\leqslant x\leqslant s\leqslant 1\\ \dfrac{1}{6}(sx^3+xs^3-3sx^2+2sx-s^3), & 当0\leqslant s\leqslant x\leqslant 1\end{cases}$$

此例中，$\lambda=-1, f(x)=x^2$，并且

$$\lambda\int_0^1 K(x,s)f(s)ds+f(x)=x^2-\frac{1}{12}x+\frac{1}{12}x^4$$

记等式右边部分为$F(x)$，则相应于（13.2.11）有

$$y(x)=\int_0^1 K_2(x,s)y(s)ds+F(x)$$

对此积分方程应用数值求积方法，仍采用二个节点的

Gauss-Legendre求积公式，有

$K_2(s_1,s_1)=K_2(s_2,s_2)=0.00926$

$K_2(s_1,s_2)=K_2(s_2,s_1)=0.00678$

$F(s_1)=0.02722$，$F(s_2)=0.58854$

由此得线性代数方程组

$$\begin{cases} 0.99537\widetilde{y_1}-0.00339\widetilde{y_2}=0.02722 \\ -0.00339\widetilde{y_1}+0.99537\widetilde{y_2}=0.58854 \end{cases}$$

解这个方程组，得出

$\widetilde{y_1}=0.02936$ ，$\widetilde{y_2}=0.57138$

这个解与积分方程的精确解y在求积的节点上的值相比较有

$y(s_1)-\widetilde{y_1}=-0.00016$，$y(s_2)-\widetilde{y_2}=0.00049$

可以看出，这是相当精确的。

13.3 近似退化核替代法

对于第二类Fredholm积分方程（13.1.1）

$$y(x)=\lambda\int_a^b K(x,s)y(s)ds+f(x)$$

如果其核K可以表示为

$$K(x,s)=\sum_{i=1}^{n}\alpha_i(x)\beta_i(s)$$

则称其核是退化的，其中假定函数系$\{\alpha_i\}$是线性无关的，$\{\beta_i\}$也是线性无关的。

近似退化核替代法(Method of Replacing Approximately the Kernel by a Degenerate One)是基于退化核积分方程求其精确解的方法。设积分方程（13.1.1）的核K可以近似地用一个退化核来替代，即

13.3.1 $K(x,s)\approx\sum_{i=1}^{n}\alpha_i(x)\beta_i(s)$

并设寻求的（13.1.1）的近似解$\widetilde{y}$形如

13.3.2 $\widetilde{y}(x)=\lambda\sum_{i=1}^{n}c_i\alpha_i(x)+f(x)$

其中

13.3.3 $c_i=\int_a^b\beta_i(s)\widetilde{y}(s)ds$

把表达式（13.3.2）代入（13.3.3），得到

$$c_i=\int_a^b\beta_i(s)f(s)ds+\lambda\int_a^b\beta_i(s)\sum_{j=1}^{n}c_j\alpha_j(s)ds\quad(i=1,2,\cdots,n)$$

引入记号

13.3.4 $f_i=\int_a^b\beta_i(s)f(s)ds$，$A_{ij}=\int_a^b\alpha_j(s)\beta_i(s)ds$

从而得出

13.3.5 $c_i-\lambda\sum_{j=1}^{n}c_jA_{ij}=f_i\quad(i=1,2,\cdots,n)$

这是关于c_i的线性代数方程组，解这个方程组就可得到c_1，c_2，$\cdots c_n$，把这些数代入表达式(13.3.2)就得到第二类Fredholm积分方程（13.1.1）的近似解。

通常，取K的按某一标准正交函数系$\{\varphi_k\}$的Fourier级数的部分和或K的Taylor级数的部分和作为近似退化核。

13.3.6 **例** 用近似退化核替代法求积分方程

$$y(x)=\int_0^1\sinh(xs)y(s)ds+1-x^2$$

的近似解。

解 对积分方程的核$K(x,s)=\sinh(xs)$，用其Taylor

级数的前三项的部分和

$$\sinh(xs)\approx xs+\frac{(xs)^3}{3!}+\frac{(xs)^5}{5!}$$

来替代，可取

$\alpha_1(x)=x,\quad \alpha_2(x)=x^3,\quad \alpha_3(x)=x^5$

$\beta_1(s)=s,\quad \beta_2(s)=\frac{1}{3!}s^3\ ,\quad \beta_3(s)=\frac{1}{5!}s^5$

$f(x)=1-x^2$

把近似解$\widetilde{y}$取（13.3.2）的形式

$\widetilde{y}(x)=c_1x+c_2x^3+c_3x^5+1-x^2$

利用公式(13.3.4)计算出线性代数方程组的系数及右端项，为

$$f_1=\int_0^1\beta_1(s)f(s)ds=\frac{1}{4}$$

$$f_2=\int_0^1\beta_2(s)f(s)ds=\frac{1}{72}$$

$$f_3=\int_0^1\beta_3(s)f(s)ds=\frac{1}{2880}$$

$$A_{11}=\int_0^1 s^2ds=\frac{1}{3},\quad A_{12}=\int_0^1 s^4ds=\frac{1}{5}$$

$$A_{13}=\int_0^1 s^6ds=\frac{1}{7},\quad A_{21}=\int_0^1\frac{1}{3!}s^4ds=\frac{1}{30}$$

$$A_{22}=\int_0^1\frac{1}{3!}s^6ds=\frac{1}{42},\ A_{23}=\int_0^1\frac{1}{3!}s^8ds=\frac{1}{54}$$

$$A_{31}=\int_0^1 \frac{1}{5!}s^6 ds=\frac{1}{840}, \quad A_{32}=\int_0^1 \frac{1}{5!}s^8 ds=\frac{1}{1080}$$

$$A_{33}=\int_0^1 \frac{s^{10}}{5!}ds=\frac{1}{1320}$$

于是得到线性代数方程组

$$\begin{cases} c_1=\frac{1}{3}c_1+\frac{1}{5}c_2+\frac{1}{7}c_3+\frac{1}{4} \\ c_2=\frac{1}{30}c_1+\frac{1}{42}c_2+\frac{1}{54}c_3+\frac{1}{72} \\ c_3=\frac{1}{840}c_1+\frac{1}{1080}c_2+\frac{1}{1320}c_3+\frac{1}{2880} \end{cases}$$

用迭代法求解，可得$c_1=0.3833, c_2=0.0273, c_3=0.0008$，由此得近似解$\tilde{y}$

$$\tilde{y}(x)=0.3833x+0.0273x^2+0.0008x^5+1-x^2$$

13.4

13.4 矩量法

用矩量法（Moment Method）求积分方程

13.4.1 $$L(y)=y(x)-\lambda\int_a^b K(x,s)y(s)ds-f(x)=0$$

的近似解时，需要事先选择一族函数

$\psi_1, \psi_2, \cdots, \psi_n, \cdots$

使它们是线性无关并且是完备的．对（13.4.1），设其近似解为

13.4.2 $$y_n(x)=\sum_{i=1}^{n} c_i\psi_i(x)+f(x)$$

其中$\psi_1, \psi_2, \cdots, \psi_n$为上述函数族中的前$n$个函数，$c_1, c_2, \cdots, c_n$是$n$个待定系数．

$c_1, c_2, \cdots, c_n$可以由

13.4.3 $\int_a^b L(y_n)\psi_j(x)dx=0 \quad (j=1,2,\cdots,n)$

来确定。

把（13.4.3）具体写出为

13.4.4
$$\sum_{i=1}^{n} c_i\int_a^b \psi_i(x)\psi_j(x)dx-\lambda\sum_{i=1}^{n} c_i\int_a^b\int_a^b K(x,s)\psi_i(s)\psi_j(x)dsdx$$
$$=\lambda\int_a^b\int_a^b K(x,s)f(s)\psi_j(x)dsdx \qquad (j=1,2,\cdots,n)$$

求解关于$c_1,c_2,\cdots,c_n$的线性方程组（13.4.4）就得到积分方程（13.4.1）的近似解y_n。

13.5

13.5 特征值问题

用数值方法求解积分方程

13.5.1 $\int_a^b K(x,s)y(s)ds=\kappa y(x)$

的特征值κ及特征函数$y(x)$，其中a,b为有限的常数，K为给定的核。

选取适当的数值求积公式

13.5.2 $\int_a^b F(x)dx\approx\sum_{k=1}^{n} A_kF(x_k)$

代入（13.5.1）中的积分，得

$$\sum_{k=1}^{n} A_kK(x,x_k)\widetilde{y}(x_k)=\kappa^*\widetilde{y}(x)$$

其中κ^*，$\widetilde{y}$分别是积分方程(13.5.1）中的κ和y的近似值.特别地，令$x=x_1,x_2,\cdots x_n$,则得到一个代数特征值问题

13.5.3 $\sum_{k=1}^{n} A_k K(x_j,x_k)\tilde{y}(x_k)=\kappa^* \tilde{y}(x_j) \quad (j=1,2,\cdots,n)$

此式可以写成矩阵形式

13.5.4 $KD\tilde{y}=\kappa^* \tilde{y}$

其中

$$K=\begin{pmatrix} K(x_1,x_1) & K(x_1,x_2) & \cdots & K(x_1,x_n) \\ K(x_2,x_1) & K(x_2,x_2) & \cdots & K(x_2,x_n) \\ \cdots & \cdots & & \\ K(x_n,x_1) & K(x_n,x_2) & \cdots & K(x_n,x_n) \end{pmatrix}$$

$D=\mathrm{diag}(A_1,A_2,\cdots,A_n)$

$\tilde{y}=(\tilde{y}_1,\tilde{y}_2,\cdots,\tilde{y}_n)^T, \ \tilde{y}_j=\tilde{y}\ (x_j)$

如果核K是实对称的，那么可以把（13.5.4）改写为

13.5.5 $(D^{\frac{1}{2}}KD^{\frac{1}{2}})\tilde{Z}=\kappa^*\tilde{Z}$

其中 $\tilde{Z}=D^{\frac{1}{2}}\tilde{y}$.由于矩阵$D^{\frac{1}{2}}KD^{\frac{1}{2}}$是实对称矩阵，因此特征向量和特征值容易求出。

13.5.6 **例** 求积分方程

$$\int_0^1 (1-3xs)y(s)ds=\kappa y(x)$$

的特征值和相应的特征函数。

解 取求积公式（13.5.2)为梯形公式.并把它代入积分方程，那么对每个x得

13.5.7 $\frac{1}{2}\tilde{y}(0)+\frac{1}{2}(1-3x)\tilde{y}(1)=\kappa^*\tilde{y}(x)$

特别地，取$x=0,1$，则得方程组

$$\begin{pmatrix} \frac{1}{2} & \frac{1}{2} \\ \frac{1}{2} & -1 \end{pmatrix} \begin{pmatrix} \tilde{y}(0) \\ \tilde{y}(1) \end{pmatrix} = \kappa^* \begin{pmatrix} \tilde{y}(0) \\ \tilde{y}(1) \end{pmatrix}$$

容易求出特征值为$-\frac{1}{4} \pm \frac{\sqrt{13}}{4}$，相应的特征函数$\tilde{y}$可由$\tilde{y}(0)$，$\tilde{y}(1)$及（13.5.7）得到。

由于梯形公式误差较大，因此所得到的特征值及相应的特征函数误差也较大。

取$h=\frac{1}{2}$的复化梯形公式，可以得

$$\frac{1}{4}\tilde{y}(0) + \frac{1}{2}\left(1-\frac{3}{2}x\right)\tilde{y}\left(\frac{1}{2}\right) + \frac{1}{4}(1-3x)\tilde{y}(1)$$

$$= \kappa^* \tilde{y}(x)$$

取$x=0$，$\frac{1}{2}$，1，则得线性代数方程组

$$\begin{pmatrix} \frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\ \frac{1}{4} & \frac{1}{8} & -\frac{1}{8} \\ \frac{1}{4} & -\frac{1}{4} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} \tilde{y}(0) \\ \tilde{y}(\frac{1}{2}) \\ \tilde{y}(1) \end{pmatrix} = \kappa^* \begin{pmatrix} \tilde{y}(0) \\ \tilde{y}(\frac{1}{2}) \\ \tilde{y}(1) \end{pmatrix}$$

或可以写作对称形式

$$
\begin{pmatrix} \frac{1}{4} & \frac{\sqrt{2}}{4} & \frac{1}{4} \\ \frac{\sqrt{2}}{4} & \frac{1}{8} & -\frac{\sqrt{2}}{8} \\ \frac{1}{4} & -\frac{\sqrt{2}}{8} & -\frac{1}{2} \end{pmatrix} \begin{pmatrix} \widetilde{Z}(0) \\ \widetilde{Z}(\frac{1}{2}) \\ \widetilde{Z}(1) \end{pmatrix} = \kappa^* \begin{pmatrix} \widetilde{Z}(0) \\ \widetilde{Z}(\frac{1}{2}) \\ \widetilde{Z}(1) \end{pmatrix}
$$

其中

$$
(\widetilde{Z}(0), \widetilde{Z}(\tfrac{1}{2}), \widetilde{Z}(1))^T
$$

$$
=(\frac{1}{2}\widetilde{y}(0), \frac{1}{\sqrt{2}}\widetilde{y}(\frac{1}{2}), \frac{1}{2}\widetilde{y}(1))^T
$$

可以求出其特征值为 $-\frac{1}{16}\pm\frac{\sqrt{26}}{8}$，0，仿前可以求出相应的特征函数。

13.6

13.6 用多步法解第二类 Volterra积分方程

假定第二类Volterra(沃尔泰拉)积分方程(Volterra Integral Equation of Second Kind)为

13.6.1 $$y(x)=\int_0^x K(x,s)y(s)ds+f(x)$$

其中核 K 在 $R=\{(x,s)\ |\ 0\leqslant s\leqslant x\leqslant b\}$ 上连续，f 在 $[0,b]$ 上连续。这样的方程有唯一解。**多步方法** (Multi-step Method) 如下：设 $x=nh$，采用等距节点 $x_0=0, x_k=kh$ ($k=1,2,\cdots n$) 的求积公式

13.6.2 $$\int_0^x F(s)ds\approx\sum_{k=0}^{n}A_{nk}F(x_k)=\sum_{k=0}^{n}A_{nk}F(kh)$$

把公式 (13.6.2) 代入 (13.6.1) 的积分中，可以得到

$$
\widetilde{y}(nh)=\sum_{k=0}^{n}A_{nk}K(nh,kh)\widetilde{y}(kh)+f(nh)
$$

把上式改写为多步法的一般公式

13.6.3 $[1-A_{nn}K(nh,nh)]\tilde{y}(nh)$

$$=\sum_{k=0}^{n-1}A_{nk}K(nh,kh)\tilde{y}(kh)+f(nh)$$

特别地，把等距求积公式（13.6.2）取为复化梯形公式，那么公式（13.6.3）简化为

13.6.4 $[1-\frac{1}{2}hK(nh,nh)]\tilde{y}(nh)$

$$=h\sum_{k=1}^{n-1}K(nh,kh)\tilde{y}(kh)+\frac{1}{2}hK(nh,0)\tilde{y}(0)+f(nh)$$

由于从积分方程可以得到$\tilde{y}(0)=f(0)$，因此这个公式开始计算特别容易，$\tilde{y}(h),\tilde{y}(2h)$，…，$\tilde{y}(nh)$都可从公式（13.6.4）递推地求出。

13.6.5 **例** 在区间$[0,2]$上，使用$n=20$的复化梯形公式求积分方程

$$y(x)=\int_0^x\frac{1}{5}xsy(s)ds+x,\quad x>0$$

的近似解。

解 首先，由于$\tilde{y}(0)=0$，利用公式（13.6.4）有

$$[1-0.05\times\frac{1}{5}(0.1)^2]\tilde{y}(0.1)$$

$$=0.1+0.05\times\frac{1}{5}\times0.1\times0\times\tilde{y}(0)=0.1$$

因此

$\tilde{y}(0.1)=0.10001$

同样地可以得出

$\tilde{y}(0.2)=0.20012$，$\tilde{y}(0.3)=0.30057,\cdots$，

$\tilde{y}(1.9)=3.00564$，$\tilde{y}(2)=3.41463$

此例积分方程的精确解为

$$y(x)=x\exp\left(\frac{x^3}{15}\right)$$

因此可以得到误差估计

$$\tilde{y}(0.3)-y(0.3)=0.30057-0.30054=0.00003$$

$$\tilde{y}(1.9)-y(1.9)=3.00564-3.00152=0.00412$$

为了更精确地计算，必须用高阶求积公式来代替复化梯形公式，但是，使用高阶求积公式必须有一个特殊开始方法。最常用的是Day（戴依）方法，计算 $\tilde{y}(h)$，$\tilde{y}(2h)$，$\tilde{y}(3h)$的公式有：

13.6.6 $\tilde{y}(h)$

$$=\frac{1}{6}h[K(h,0)f(0)+4K(h,\frac{1}{2}h)y_{13}+K(h,h)y_{12}]+f(h)$$

其中

$$y_{11}=hK(h,0)f(0)+f(h)$$

$$y_{12}=\frac{1}{2}h[K(h,0)f(0)+K(h,h)y_{11}]+f(h)$$

$$y_{13}=\frac{1}{4}h[K(\frac{1}{2}h,0)f(0)$$

$$+K(\frac{1}{2}h,\frac{1}{2}h)(\frac{1}{2}f(0)+\frac{1}{2}y_{12})]+f(\frac{1}{2}h)$$

13.6.7 $\tilde{y}(2h)=\frac{1}{3}h[K(2h,0)f(0)+4K(2h,h)\tilde{y}(h)$

$$+K(2h,2h)y_{21}]+f(2h)$$

其中

$$y_{21}=2hK(2h,h)\tilde{y}(h)+f(2h)$$

13.6.8 $\tilde{y}(3h)=\frac{3}{8}h[K(3h,0)f(0)+3K(3h,h)\tilde{y}(h)$

$$+3K(3h,2h)\widetilde{y}(2h)+K(3h,3h)y_{31}]+f(3h)$$

其中

$$y_{31}=\frac{3}{2}h[K(3h,h)\widetilde{y}(h)+K(3h,2h)\widetilde{y}(2h)]+f(3h)$$

Day方法与复化Simpson求积公式的联合使用，改善了求解的精度.但是，由于复化Simpson求积公式仅适用于奇数个等距节点，所以，对偶数个节点的情况，可采用3/8 Simpson求积公式.

13.6.9 **例** 在区间[0,2]上使用节点距$h=0.1$的Simpson求积公式和3/8 Simpson求积公式计算积分方程

$$y(x)=\int_0^x \frac{1}{5}xsy(s)ds+x,\quad x>0$$

的近似解.

解 由于$h=0.1, K(x,s)=\frac{1}{5}xs$，所以首先采用Day方法来确定$\widetilde{y}(h)$，得

$$y_{11}=0.1$$

$$y_{12}=0.1+0.05\left[0+\frac{1}{5}(0.1)^3\right]=0.10001$$

$$y_{13}=0.05+0.025\left[0+\frac{1}{5}(0.05)^2(0+0.05)\right]\approx 0.05$$

$$\widetilde{y}(h)=\widetilde{y}(0.1)$$

$$=0.1+\frac{1}{6}(0.1)\left[0+4\times\frac{1}{5}(0.1)(0.05)(0.05)+\frac{1}{5}(0.1)(0.1)(0.10001)\right]$$

$$=0.10001$$

利用Simpson公式计算，有

$\widetilde{y}(2h)=\widetilde{y}(0.2)=0.20011$

再用3/8Simpson公式计算，有

$\widetilde{y}(3h)=\widetilde{y}(0.3)=0.30054$

在整个求解过程中，$\widetilde{y}(0.4)$，$\widetilde{y}(0.6)$，$\cdots\widetilde{y}(2.0)$是用Simpson公式计算，而$\widetilde{y}(0.5)$，$\widetilde{y}(0.7)$，$\cdots\widetilde{y}(1.9)$则皆用3/8 Simpson公式进行计算。

为进行比较，在表（13.6.10）中列出了本例的数值结果，积分方程的精确解以及用复化梯形求积得到的结果。

表13.6.10

x_k	精确解	梯形	Simpson
0.0	0.00000	0.00000	0.00000
0.1	0.10001	0.10001	0.10001
0.2	0.20011	0.20012	0.20011
0.3	0.30054	0.30057	0.30054
0.6	0.60870	0.60883	0.60870
0.9	0.94482	0.94514	0.94482
1.2	1.34652	1.34720	1.34652
1.5	1.87849	1.87993	1.87849
1.8	2.65538	2.65582	2.65538
1.9	3.00152	3.00564	3.00154
2.0	3.40921	3.41463	3.40922

虽然没有把全部计算结果列出，但可以看出，采用Simpson公式比梯形公式要精确得多。

13.7 用Runge-Kutta型方法解第二类Volterra积分方程

积分方程

$$y(x)=\int_0^x (s-x)y(s)ds+\alpha+\beta x$$

等价于常微分方程的初值问题

$y''(x)+y(x)=0,\ y(0)=\alpha,\ y'(0)=\beta$

由此可以采用类似于求解常微分方程的办法来求解积分方程。采用高阶求积公式的多步法必须使用特别的开始步骤，这是很不方便的。此外，改变步长也不容易。采用Runge-Kutta型方法来求解积分方程

$$y(x)=\int_0^x k(x,s)y(s)ds+f(x)$$

可以克服多步法的缺点。用Runge-Kutta型方法来求解，有不同的形式，例如，使用Pouzet（布泽）公式。令

13.7.1 $$\begin{cases} y_n^{(0)}=y_{n-1}^{(4)}\ (y_0^{(0)}=f(x_0)) \\ y_n^{(1)}=F_n(x_{n+\frac{1}{2}})+\frac{1}{2}hK(x_{n+\frac{1}{2}},x_n)y_n^{(0)} \\ y_n^{(2)}=F_n(x_{n+\frac{1}{2}})+\frac{1}{2}hK(x_{n+\frac{1}{2}},x_{n+\frac{1}{2}})y_n^{(1)} \\ y_n^{(3)}=F_n(x_{n+1})+hK(x_{n+1},x_{n+\frac{1}{2}})y_n^{(2)} \end{cases}$$

其中

$$F_n(x)=f(x)+\frac{1}{6}h\sum_{j=0}^{n-1}\{K(x,x_j)y_j^{(0)}$$
$$+2K(x,x_{j+\frac{1}{2}})y_j^{(1)}+2K(x,x_{j+\frac{1}{2}})y_j^{(2)}$$
$$+K(x,x_{j+1})y_j^{(3)}\}$$

$F_0(x)=f(x)$

设

13.7.2 $$y_n^{(4)}=F_n(x_{n+1})+\frac{1}{6}h\{K(x_{n+1},x_n)y_n^{(0)}$$

$$+2K(x_{n+1},x_{n+\frac{1}{2}})y_n^{(1)}+2K(x_{n+1},x_{n+\frac{1}{2}})y_n^{(2)}$$

$$+K(x_{n+1},x_{n+1})y_n^{(3)}\}$$

那么第二类Volterra积分方程在$x=x_{n+1}$的近似解$\widetilde{y}$可以表示为

13.7.3 $\widetilde{y}(x_{n+1})=y_n^{(4)}$

13.7.4 **例** 用Runge-Kutta型方法求积分方程

$$y(x)=\int_0^x \frac{1}{5}xsy(s)ds+x \qquad x>0$$

的解在$x=x_k=kh$（$h=0.1,k=0,1,2,\cdots 20$）的近似值。

解 按公式计算，得

$$y_0^{(0)}=0$$

$$y_0^{(1)}=0.05+0.05\left(\frac{1}{5}\right)(0.05)(0)(0)=0.05$$

$$y_0^{(2)}=0.05+0.05\left(\frac{1}{5}\right)(0.05)^3\approx 0.05$$

$$y_0^{(3)}=0.1+0.1\left(\frac{1}{5}\right)(0.1)(0.05)^2\approx 0.10001$$

于是有

$$\widetilde{y}(0.1)\approx y_0^{(4)}=0.1+\frac{1}{6}(0.1)\left\{\frac{1}{5}(0.1)(0)(0)\right.$$

$$+2\left(\frac{1}{5}\right)(0.1)(0.05)(0.05)+2\left(\frac{1}{5}\right)(0.1)(0.05)(0.05)$$

$$\left.+\frac{1}{5}(0.1)(0.1)(0.10001)\right\}=0.10001$$

类似地

$$y_1^{(0)}=0.10001,y_1^{(1)}=0.15002,\ y_1^{(2)}=0.15004$$

$$y_1^{(3)}=0.20010$$

于是有

$$\widetilde{y}(0.2)\approx y_1^{(4)}=0.20010$$

其余近似值仿此可求出.在一些点上计算的结果列于表(13.7.5)中.

13.7.5 表

x_k	0.1	0.4	0.7	1.0
计算解	0.10001	0.40171	0.71619	1.06894
精确解	0.10001	0.40171	0.71619	1.06894
x_k	1.3	1.6	1.9	2.0
计算解	1.50506	2.10238	3.00152	3.40920
精确解	1.50506	2.10238	3.00152	3.40921

由数值结果可以看出，这个方法相当精确，但是在计算过程中，函数值计算次数多，耗时也多.

附　　录

本附录共有二十个程序，其中，插值与逼近三个，方程求根一个，数值积分二个，线性方程组的直接解法三个，线性方程组的迭代法与加速方法四个，矩阵特征值与特征向量计算二个，非线性方程组解法三个，常微分方程和方程组的初值问题数值解法二个。

全部程序用FORTRAN语言编写，在Universe-68(宇宙-68)微型计算机上调试通过，并且分别用计算实例验算正确。每个程序由二部分组成：在前的是一个或若干个子程序；在后的是一个主程序（调试程序）。用DATA语句输入原始数据。输入、输出的设备号为*，对于其它型号的计算机，应当改用其相应的输入、输出设备号。注释行和继续行按FORTRAN语言的规定书写。

每个附录包括该程序的功能、使用说明、例题、程序和计算结果共四部分。计算方法详见本手册的相应章节。

1.　三次样条插值

1.1　功能

本程序利用已知的n个插值节点$x_1<x_2<\cdots<x_n$以及对应的函数值$y_1,y_2,\cdots,y_n$，采用Lagrange（拉格朗日）三次样条插值函数计算一组插值点上的函数值。

计算由二个子程序组成，先调用子程序SPLINE，用它来求解三对角方程组，从而得到样条函数的系数。然后调用子程序VALSPL，用它来计算一组插值点$z_1,z_2,\cdots z_m$上的样条函数值，计算一个值调用一次。

1.2　使用说明

1° 子程序调用语句

CALL SPLINE (X,FX,N1,B,A,SD,SU)

CALL VALSPL (X,A,N1,Z,SVAL)

2° 虚元和工作单元说明

（1）子程序SPLINE的虚元说明

输入参数：

N1 整变量，插值节点数

X N1个元素的一维实数组，存放插值节点

FX N1个元素的一维实数组，存放相应插值节点的函数值

输入兼输出参数：

B N1个元素的一维实数组，开始存放方程组的右端项，最后存放方程组的解向量

输出参数：

A 4 • N1个元素的二维实数组，存放样条函数中的系数

工作单元：

SD,SU N1个元素的一维实数组，分别存放三对角矩阵的主对角线元素和次对角线元素

（2）子程序VALSPL的虚元说明

输入参数：

N1 整变量，插值节点数

X N1个元素的一维实数组，存放插值节点

A 4 • N1个元素的二维实数组，存放样条函数中的系数

Z 实变量，插值点

输出参数：

SVAL 实变量，Z点的样条函数值

1.3 例题

已知插值节点

$$x_i = -1 + 2(i-1)N \qquad (i = 1, 2, \cdots, N+1) \tag{1}$$

以及相应的函数值

$$f(x_i)=1/(1+25x_i^2) \quad (i=1,2,\cdots,N+1) \qquad ②$$

求最大的插值误差

$$\max_{1\leq j\leq 151} |f(z_j)-S_L(z_j)| \qquad ③$$

其中

$$z_j=-1+2(j-1)/150 \quad (j=1,2,\cdots,151) \qquad ④$$

是插值点，$S_L(z_j)$ 是相应的样条函数值。

对于$N=8,16,\cdots,80$ 分别计算③。

1.4 程序和计算结果

```
      SUBROUTINE SPLINE (X, FX, N1, B, A, SD, SU)
      REAL X(N1), FX(N1), B(N1), SD(N1), SU(N1),
     +A(4, N1)
C  DETERMINE THE RIGHT HAND SIDE
      N=N1-1
      DO 2 J=1, N
      B(J)=(FX(J+1)-FX(J))/(X(J+1)-X(J))
    2 CONTINUE
      DO 3 J=2, N
      JJ=N-J+2
      B(JJ)=6. *(B(JJ)-B(JJ-1))
    3 CONTINUE
C  ENDPOINT CONDITIONS
      W=B(2)/(X(3)-X(1))
      Z=B(3)/(X(4)-X(2))
      Z=(Z-W)/(X(4)-X(1))
      B(1)=(X(2)-X(1))*((X(3)-X(1))*Z-W)
      W=B(N-1)/(X(N)-X(N-2))
      Z=B(N)/(X(N1)-X(N-1))
      W=(Z-W)/(X(N1)-X(N-2))
      B(N1)=(X(N1)-X(N))*(Z+(X(N1)-X(N-1))*W)
C  SET UP COEFFICIENT MATRIX
      SD(1)=2.*(X(2)-X(1))
      SU(1)=SD(1)/2.
      DO 4 J=2, N
      SU(J)=X(J+1)-X(J)
      SD(J)=2.*(X(J+1)-X(J-1))
```

```
    4 CONTINUE
      SD(N1)=2.*(X(N1)-X(N))
C  TRIDIAGONAL ALGORRITHM FOR SOLUTION
C  OF LINEAR SYSTEM CORRESPONDING TO
C  THE CUBIC SPLINE
      DO 5 J=2, N1
      Q=SU(J-1)/SD(J-1)
      SD(J)=SD(J)-SU(J-1)*Q
      B(J)=B(J)-Q*B(J-1)
    5 CONTINUE
      B(N1)=B(N1)/SD(N1)
      DO 6 K=1, N
      KK=N1-K
      B(KK)=(B(KK)-SU(KK)*B(KK+1))/SD(KK)
    6 CONTINUE
      DO 7 J=1, N
      DELTA=X(J+1)-X(J)
      A(4, J)=(B(J+1)-B(J))/(6.*DELTA)
      A(3, J)=B(J)/2.
      A(2, J)=(FX(J+1)-FX(J))/DELTA
      A(2, J)=A(2, J)-DELTA*(B(J+1)+2.*B(J))/6.
      A(1, J)=FX(J)
    7 CONTINUE
      RETURN
      END

      SUBROUTINE VALSPL (X, A, N1, Z, SVAL)
      DIMENSION X(N1), A(4, N1)
      INDX=1
      N=N1-2
C  USE LINEAR SEARCH TO FIND SUBINTERVAL INDEX
      DO 1 J=1, N
      IF(Z. LE. X(J+1)) GOTO 2
      INDX=INDX+1
    1 CONTINUE
    2 K=INDX
      W=Z-X(K)
      SVAL=A(1, K)+W*(A(2, K)+W*(A(3,K)+W*A(4,K)))
      RETURN
      END

      REAL FUNCTION F(X)
```

```
      F=1./(1.+25.*X**2)
      RETURN
      END

      DIMENSION A(4, 81), X(81), FX(81), SU(81),
     +SD(81), B(81)
      DATA AA, BB/-1., 1./
      WRITE(*,99)
   99 FORMAT(8X,' N', 8X,' MAX ERROR')
      DO 3 N=8, 80, 8
      N1=N+1
      H=(BB-AA)/FLOAT(N)
      DO 1 K=1, N1
      X(K)=AA+FLOAT(K-1)*H
      FX(K)=F(X(K))
    1 CONTINUE
      CALL SPLINE(X, FX, N1, B, A, SD, SU)
      ERR=0.
      DO 2 J=1, 151
      P=-1. +2. *FLOAT(J-1)/150.
      CALL VALSPL(X, A, N1, P, SVAL)
      DIFF=ABS(SVAL-F(P))
      IF(DIFF.GT.ERR) ERR=DIFF
    2 CONTINUE
      WRITE(*, 101)N, ERR
  101 FORMAT(7X, I3, 3X, E14.7)
    3 CONTINUE
      STOP
      END
```

```
 N        MAX ERROR
 8      .5584902E-01
16      .3741741E-02
24      .1917601E-02
32      .6184578E-03
40      .2779365E-03
48      .1212358E-03
56      .6912781E-04
64      .3784895E-04
72      .2712756E-04
80      .2053380E-04
```

2. 有理函数插值

2.1 功能

本程序对于已知的 n 个节点 $x_1, x_2, \cdots, x_n$ 及相应的函数值 y_1, $y_2, \cdots, y_n$，采用连分式函数和有理分式函数，对一元函数进行有理插值。

2.2 使用说明

1° 子程序调用语句

CALL CONFR (N,X,A,B,D,P,Q,N1,T,Y)

2° 虚元说明

输入参数：

N 整变量，节点个数

N1 整变量，N1=[N/2]+1

T 实变量，插值点

X 2*N个元素的二维实数组，存放插值节点 x_i 和对应的函数值 y_i ($i=1,2,\cdots,N$)，存放顺序如下：

$x_1, y_1, x_2, y_2, \cdots, x_n, y_n$

输出参数：

Y 实变量，插值结果

A N个元素的一维实数组，存放连分式系数

B N1个元素的一维实数组，存放有理式分子多项式系数

D N1个元素的一维实数组，存放有理式分母多项式系数

工作单元：

P,Q 均为N1个元素的一维实数组

2.3 例题

已知 e^x 在 $x=0$，±0.1，±0.2，±0.3，±0.4，±0.5，±0.6 处的函数值，求 $x=\pm0.05$，±0.15，±0.25，±0.35，±0.45，

±0.55时的函数值.分[0,0.6]和[−0.6，0]两种情况计算

x	−0.6	−0.5	−0.4	−0.3	−0.2	−0.1	0.0
e^x	0.548812	0.606531	0.670320	0.740818	0.818731	0.904837	1.0

x	0.0	0.1	0.2	0.3	0.4	0.5	0.6
e^x	1.0	1.051709	1.221403	1.349859	1.491825	1.648721	1.822119

2.4 程序和计算结果

```
      SUBROUTINE CONFR(N, X, A, B, D, P, Q, N1, T, Y)
      DIMENSION X(2, N), A(N), B(N1), D(N1), P(N1), Q(N1)
      A(1)=X(2, 1)
      DO 2 J=2, N
      AA=X(2, J)
      XX=X(1, J)
      J1=J-1
      DO 1 I=1, J1
    1 AA=(XX-X(1, I))/(AA-A(I))
    2 A(J)=AA
      K=1
      P(1)=1.0
      Q(1)=A(1)
    3 DO 4 J=2, N1
      P(J)=0.0
    4 Q(J)=0.0
      DO 6 I=2, N
      I2=I/2+0.1
      DO 5 J=1, I2
      J1=I2-J+1
      Z=A(I)*Q(J1+1)-X(1, I-1)*P(J1+1)+P(J1)
      P(J1+1)=Q(J1+1)
    5 Q(J1+1)=Z
      Z=A(I)*Q(1)-X(1, I-1)*P(1)
      P(1)=Q(1)
    6 Q(1)=Z
      GO TO (7, 9), K
    7 DO 8 J=1, N1
```

```
 8  B(J)=Q(J)
    K=2
    P(1)=0.0
    Q(1)=1.0
    GO TO 3
 9  DO 10 J=1, N1
10  D(J)=Q(J)
    Y1=B(1)
    Y2=D(1)
    DO 11 J=2, N1
    Y1=Y1+B(J)*T**(J-1)
11  Y2=Y2+D(J)*T**(J-1)
    Y=Y1/Y2
    RETURN
    END
```

```
    DIMENSION VV(2, 7),UU(2, 7),B(4),D(4),P(4), Q(4), C(7)
    DATA VV/-.6, .548812, -.5, .606531, -.4, .67032, -.3,
   +.740818, -.2, .818731, -.1, .904837, 0.0, 1.0/, UU/0.0,
   +1.0, .1, 1.051709, .2, 1.221403, .3, 1.349859, .4,
   +1.491825, .5, 1.648721, .6, 1.822119/
    X=0.05
    T=-0.05
    WRITE(*, 40)
40  FORMAT(11X, 1HX, 7X, 3HEXP, 9X, 1HX, 7X, 3HEXP)
    DO 30 I=1, 6
    CALL CONFR(7, VV, C, B, D, P, Q, 4, X, Y)
    CALL CONFR(7, UU, C, B, D, P, Q, 4, T, Z)
    WRITE(*, 20)X, Y, T, Z
20  FORMAT(10X, 4F10.6)
    X=X+0.1
30  T=T-0.1
    STOP
    END
```

X	EXP	X	EXP
.050000	1.0512711	-.050000	.9512236
.150000	1.1618342	-.150000	.86070835
.250000	1.2840255	-.250000	.77880074
.350000	1.4190679	-.350000	.70468788
.450000	1.5683123	-.450000	.63762832
.550000	1.7332525	-.550000	.57695022

3. 正交多项式曲线拟合

3.1 功能

本程序对于已知的n个节点$x_1, x_2, \cdots, x_n$及相应的函数值$y_1, y_2, \cdots, y_n$，采用正交多项式求得在最小二乘意义下的最佳逼近广义多项式，以及等价的幂多项式，并能自动选取多项式的"最佳"次数。

3.2 使用说明

1° 子程序调用语句

```
 CALL LSFIT (N,K,X,W,SI,P,AF,BT,S,
+CTP,CPS,CP,CLP,PL,TP,L)
```

2° 虚元说明

输入参数：

N 整变量，列表函数数据点(x, y)的个数

K 整变量，为所要逼近的多项式的最高次数加1

X 2*N个元素的二维实数组，存放自变量$x_1, x_2, \cdots, x_n$及其对应的函数值$y_1, y_2, \cdots, y_n$，存放顺序为：$x_1, y_1, x_2, y_2, \cdots, x_n, y_n$。要求$x_{i-1} < x_i$，$i = 1, 2, \cdots, n$

W N个元素的一维实数组，存放给定的数据点上y_i值的权W_i，$i = 1, 2, \cdots, n$

L 整变量，控制程序走向的信息．当L=1时，用K-1次多项式逼近；当L=0时，自动选取逼近多项式的"最佳"次数

输出参数：

SI K个元素的一维实数组，存放平均平方误差

P K个元素的一维实数组，存放幂多项式系数

S K个元素的一维实数组，存放广义多项式的系数

工作单元：

AF K个元素的一维数组，存放α_j $(j = 1, 2, \cdots, K-1)$

BT　K个元素的一维数组，存放β_j（$j=1,2,\cdots,K$）

CTP,CPS,CP,CLP,PL,TP均为K个元素的一维数组

3.3　例题

已知e^x在$x_i=-1+0.1i$（$i=0,1,\cdots,20$）上的值，求次数≤9的逼近多项式，取权$W_i=1$（$i=1,2,\cdots,n$），K=7.

3.4　程序和计算结果

```
      SUBROUTINE LSFIT(N,K,X,W,SI,P,AF,BT,S,CTP,CPS,CP,
     +CLP, PL, TP, L)
      DIMENSION X(2, N), W(N), SI(K), P(K), AF(K), BT(K)
     +CPS(K), CP(K), CLP(N), PL(N), TP(N), S(K), CTP(K)
      DO 1 I=1, K
    1 CP(I)=0.0
      SIMIN=0.0
      LS=1
      BT(1)=0.0
      CLP(1)=0.0
      CLP(2)=0.0
      DE=0.0
      CM=0.0
      CTP(1)=1.0
      TW=0.0
      LC=0
      DO 2 I=1, N
      TP(I)=1.0
      PL(I)=0.0
      DE=DE+W(I)*X(2, I)**2
      CM=CM+W(I)*X(2, I)
    2 TW=TW+W(I)
      S(1)=CM/TW
      CP(1)=S(1)
      DE=DE-S(1)*CM
      SI(1)=DE/(N-1)
      A=4.0/(X(1, N)-X(1, 1))
      B=-2.0-A*X(1, 1)
      DO 3 I=1, N
    3 X(1, I)=A*X(1, I)+B
      K1=K-1
```

```
      DO 12 I=1, K1
      DU=0.0
      DO 4 J=1, N
 4    DU=DU+W(J)*X(1, J)*TP(J)**2
      AF(I+1)=-DU/TW
      WL=TW
      TW=0.0
      CM=0.0
      DO 5 J=1, N
      DU=BT(I)*PL(J)
      PL(J)=TP(J)
      TP(J)=(X(1, J)-AF(I+1))*TP(J)-DU
      TW=TW+W(J)*TP(J)**2
 5    CM=CM+W(J)*X(2, J)*TP(J)
      BT(I+1)=TW/WL
      S(I+1)=CM/TW
      DE=DE-S(I+1)*CM
      SI(I+1)=DE/(N-I-1)
      IF(L. EQ. 1) GO TO 9
      IF(LC. EQ. 1) GO TO 12
      IF(LS. EQ. 0) GO TO 8
      IF(SI(I+1). LT. SI(I)) GO TO 9
      LS=0
      SIMIN=SI(I)
      DO 6 J=1, K
 6    CPS(J)=CP(J)
      GO TO 9
 8    IF(SI(I+1). LT. 0.6*SIMIN)LC=1
 9    DO 10 J=1, I
      DU=CLP(J+1)*BT(I)
      CLP(J+1)=CTP(J)
      CTP(J)=CLP(J)-AF(I+1)*CTP(J)-DU
 10   CP(J)=CP(J)+S(I+1)*CTP(J)
      CP(I+1)=S(I+1)
      CTP(I+1)=1.0
      CLP(I+2)=0.0
      IF(LC. EQ. 1) GO TO 12
      IF(LS. EQ. 1) GO TO 12
      IF(I. NE. K-1) GO TO 12
      DO 11 J=1, K
 11   CP(J)=CPS(J)
 12   CONTINUE
      CALL POLYX(A, B, CP, P, K, CLP, PL)
```

```
      RETURN
      END


      SUBROUTINE POLYX (A, B, C, D, N, Z, W)
      DIMENSION C(N), D(N), Z(N), W(N)
      W(1)=1.0
      Z(1)=1.0
      D(1)=C(1)
      DO 1 I=2, N
      W(I)=1.0
      Z(I)=B * Z(I-1)
    1 D(1)=D(1)+C(I) * Z(I)
      DO 2 J=2, N
      W(1)=W(1) * A
      D(J)=C(J) * W(1)
      K=2
      J1=J+1
      DO 2 I=J1, N
      W(K)=A * W(K)+W(K-1)
      D(J)=D(J)+C(I) * W(K) * Z(K)
    2 K=K+1
      RETURN
      END


      DIMENSION X(2.21), W(21), SI(10), P(10), AF(10),
     +BT(10), S(10), CTP(10), PL(21), TP(21), CLP(21),
     +CPS(10), CP(10)
      DO 110 I=1,21
      X(1, I)=-1.0+0.1 * (I-1)
      X(2, I)=EXP(X(1, I))
  110 CONTINUE
      DO 100 I=1,21
  100 W(I)=1.0
      K=7
      CALL LSFIT(21,K,X,W,SI,P,AF,BT,S,CTP,CPS,CP,
     +CLP, PL, TP, 1)
      WRITE( * , 104)
  104 FORMAT (17X, 2HAF, 14X, 2HBT, 14X, 2HSI)
      DO 222 I=1, 10
      WRITE( * , 105) AF(I), BT(I), SI(I)
```

```
105  FORMAT (8X, 3E14.7)
222  CONTINUE
     WRITE (*, 106)
106  FORMAT (17X, 1HS, 14X, 1HP)
     DO 112 I=1, 10
     WRITE (*, 107) S(I), P(I)
107  FORMAT (8X, 2E14.7)
112  CONTINUE
     STOP
     END
```

AF	BT	SI
.0000000E+00	.0000000E+00	.5115358E+00
.0000000E+00	.1466667E+01	.3549586E−01
−.7740862E−07	.1165333E+01	.1101845E−02
−.1859933E−06	.1110857E+01	.1902550E−04
−.4305400E−06	.1079365E+01	.1011394E−06
−.1462570E−05	.1050505E+01	−.1146620E−06
−.4387709E−05	.1019580E+01	−.1245824E−06
.0000000E+00	.0000000E+00	.0000000E+00
.0000000E+00	.0000000E+00	.0000000E+00
.0000000E+00	.0000000E+00	.0000000E+00

S	P
.1193652E+01	.9999997E+00
.5570182E+00	.1000023E+01
.1350466E+00	.5000164E+00
.2212068E−01	.1664819E+00
.2734573E−02	.4159914E−01
.2717320E−03	.8695430E−02
.2292412E−04	.1467144E−02
.0000000E+00	.0000000E+00
.0000000E+00	.0000000E+00
.0000000E+00	0000000E+00

4. 二分法求方程的根

4.1 功能

本程序用二分法求方程

$f(x)=0$

在区间 (a,b) 上的一个根，其中$f(\cdot)$是$[a,b]$上的连续函数。

4.2 使用说明

1° 子程序调用语句

CALL BYSEKT (A,B,XMID,TOL)

2° 虚元说明

输入参数：

A 实变量，区间的左端点

B 实变量，区间的右端点，B>A

TOL 实变量，根的允许绝对误差限

输出参数：

X 实变量，方程的根

3° 附注

在调用本程序的同时需要另外编制一个函数子程序$F(x)$，用来计算函数值$f(x)$。

4.3 例题

求方程$2x^3-5x-1=0$在区间[1,2]内的根。

4.4 程序和计算结果

```
      SUBROUTINE BYSEKT (A, B, XMID, TOL)
      WRITE(*, 400)
  400 FORMAT(5X, 3X, 'LEFT ENDPOINT',
     +3X,' RIGHT ENDPOINT')
      FA=F(A)
    1 XMID=(A+B)/2.
      FMID=F(XMID)
      TEST=FA*FMID
      IF(TEST. LE. 0.) GOTO 2
      A=XMID
      FA=FMID
      GOTO 3
    2 B=XMID
    3 IF((B-A). LE. TOL) GOTO 4
      WRITE (*, 500) A, B
  500 FORMAT (5X, 2(2X, E14.7))
```

```
      GOTO 1
    4 XMID=(B+A)/2.
      RETURN
      END


      REAL FUNCTION F(X)
      F=2.*X**3-5.*X-1.
      RETURN
      END
      DATA A, B, TOL/1., 2., 1.E-5/
      CALL BYSEKT (A, B, X, TOL)
      ERROR=(B-A)/2.
      WRITE(*, 101)X, ERROR
  101 FORMAT(5X,' ROOT=', 2X, F8.6,
     +', ERROR=', E14.7)
      STOP
      END
```

```
   LEFT ENDPOINT         RIGHT ENDPOINT
     .1500000E+01          .2000000E+01
     .1500000E+01          .1750000E+01
     .1625000E+01           1750000E+01
     .1625000E+01          .1687500E+01
     .1656250E+01          .1687500E+01
     .1671875E+01          .1687500E+01
     .1671875E+01          .1679687E+01
     .1671875E+01          .1675781E+01
     .1671875E+01          .1673828E+01
     .1672852E+01          .1673828E+01
     .1672852E+01          .1673340E+01
     .1672852E+01          .1673096E+01
     .1672974E+01          .1673096E+01
     .1672974E+01          .1673035E+01
     .1672974E+01          .1673004E+01
     .1672974E+01          .1672989E+01
ROOT=     1.672985,   ERROR=     .3814697E-05
```

5. Romberg求积

5.1 功能

本程序利用Romberg（龙贝格）方法计算定积分

$$I=\int_a^b f(x)dx$$

的近似值。

5.2 使用说明

1° 子程序调用语句

CALL TRAP (A,B,EPS,VALUE)

2° 虚元说明

输入参数：

A 实变量，积分下限

B 实变量，积分上限

EPS 实变量，允许的相对误差限

输出参数：

VALUE 实变量，积分结果

3° 附注

在调用本程序的同时，需要编制一个函数子程序$F(x)$，用来计算被积函数$f(x)$的值。

5.3 例题

计算定积分$I=\int_{-1}^{1}\frac{1}{1+x^2}dx$的近似值。允许相对误差$10^{-5}$。

5.4 程序和计算结果

```
      SUBROUTINE TRAP(A, B, EPS, VALUE)
      DIMENSION VI(10, 10)
C     SETUP THE INITIAL VALUES, N THE NUMBER OF STEPS,
C     NI THE NUMBER OF ITERATIONS, H THE STEP SIZE.
      N=2
      NI=2
```

```
      H=(B-A)/N
      NM1=N-1
      VALUE=F(A)+F(B)
C     VI(I, J) IS THE TABLE OF VALUES.
C     VI(I, 1) ARE GIVEN BY TRAPEZIUM RULE
      VI(1, 1)=VALUE/2.
      VALUE=VALUE+2. *F(A+H)
      VI(2, 1)=VALUE/4.
      IDIV=4
C     TO FORM THE OTHER COLUMNS FROM THE FIRST
    9 IFAC=4
      DO 20 J=2, NI
      I=NI+1-J
   21 VI(I, J)=(IFAC*VI(I+1, J-1)-VI(I, J-1))/(IFAC-1)
   20 IFAC=IFAC*4
      ACC=ABS(EPS*VI(1, NI))
C     IS THE ESTIMATE ACCURATE ENOUGH?
      IF(ABS(VI(1, NI)-VI(2, NI-1)). LE. ACC) GOTO 14
      NI=NI+1
      N=N*2
C     ARE THERE MORE ITERATIONS ALLOWED?
      IF(NI. GT. 10) GOTO 16
      H=H/2
      NM1=N-1
      IDIV=IDIV*2
      DO 13 K=1, N, 2
   13 VALUE=VALUE+2*F(A+K*H)
      VI(NI, 1)=VALUE/IDIV
      GOTO 9
   16 WRITE(*, 17)
   17 FORMAT (10X,' TERMINATED AFTER 10 ITERATIONS')
C     CALCULATE THE FINAL VALUE OF THE INTEGRAL
   14 VALUE=VI(1, NI)*(B-A)
      RETURN
      END


C     THE FUNCTION F(X)
      FUNCTION F(X)
      F=1./(1.+X*X)
      RETURN
      END
```

```
C     INPUT THE LIMITS OF INTEGRATION AND THE
C     DESIRED ACCURACY
      DATA A, B, EPS/-1. 0, 1.0, 1E-6/
C     CALL THE SUBROUTINE TO EVALUATE THE INTEGRAL.
      CALL TRAP (A, B, EPS, RESULT)
      WRITE (*, 10) RESULT, EPS
C     MODIFY F20.6, D20.6 IF NECESSARY
   10 FORMAT (10X,' VALUE OF INTEGRAL=', F20.6, /
     +10X,' TO AN ACCURACY', D20.6)
      WRITE (*, 20) A, B
   20 FORMAT (10X,' THE LIMITS ARE', 2F10.4)
      STOP
      END
```

```
VALUE OF INTEGRAL=              1.570796
TO AN ACCURACY               .100000D-05
THE LIMITS ARE         -1.0000      1.0000
```

. Gauss求积和Robinson方法

6.1 功能

本程序采用 Gauss（高斯）三点求积公式和Robinson方法计算定积分

$$I=\int_a^\infty f(x)dx$$

的近似值。

6.2 使用说明

1° 子程序调用语句

CALL ROBSON(A, B, VALUE, EPS, MORE)

2° 虚元说明

输入参数：

A　　实变量，积分下限

B　　实变量，积分上限

EPS　实变量，积分允许误差

输出参数：

VALUE　实变量，积分结果

MORE　逻辑变量，如果从主程序输出的MORE其内容是TRUE，表示需要增加迭代次数

6.3　例题

计算定积分$I=\int_0^{\pi}\sin x dx$的近似值，要求误差不超过10^{-5}。

6.4　程序和计算结果

```
C     THE SUBROUTINE USING ROBINSON'S METHOD
      SUBROUTINE ROBSON(A, B, VALUE, EPS, MORE)
      LOGICAL MORE
      DIMENSION TREE(15, 3), ROOT(3), COEFF(3)
C     INCREASE TREE(15, 3) TO TREE(127, 3)  IF REQUIRED
      MORE=.FALSE.
      TREE(1, 1)=(B-A)/2.
      TREE(1, 2)=(B+A)/2.
      DO 10 I=2, 15
   10 TREE(I, 1)=0.5
      DO 11 I=2, 14, 2
      TREE(I, 2)=0.5
   11 TREE (I+1, 2) =-0.5
C     SETUP THE COEFFICIENTS AND THE ROOTS
      ROOT (1) =-SQRT(0.6)
      ROOT (2) =0.
      ROOT (3) =-ROOT(1)
      COEFF (1)=5.0/9.
      COEFF (2)=8.0/9.
      COEFF (3)=COEFF(1)
      N=1
      M1=1
      VALUE=0.
C     SUCCESSIVELY REDUCE THE SIZE OF THE RANGES
C     INCREASE THE CONSTANT 4 TO 7 FOR EXAMPLE
C     IF NECESSARY
      DO 20 N=2, 4
```

```
      M1=M1*2
      M2=2*M1-1
      SUM=0.
      C=TREE(1,1)/M1
      DO 30 I=M1,M2
      SUM1=0.
      DO 40 K=1,3
      J=1
      ARG=ROOT(K)
   41 ARG=TREE(J,1)*ARG+TREE(J,2)
      J=J/2.
      IF(J.GT.0)GOTO 41
   40 SUM1=SUM1+F(ARG)*COEFF(K)
      TREE(I,3)=SUM1*C
C     FORM THE SUM OF THE INDIVIDUAL CONTRIBUTIONS
   30 SUM=SUM+TREE(I,3)
      N1=N-1
      WRITE(*,2)N1,SUM
    2 FORMAT(10X,'INTERMEDIATE VALUE NO',I5,'=',F12.7)
C     MODIFY F12.7 IF NECESSARY
      IF(ABS(VALUE-SUM).LE.EPS)GOTO 99
      VALUE=SUM
   20 CONTINUE
      MORE=.TRUE.
   99 RETURN
      END


C     THE FUNCTION
      FUNCTION F(X)
      F=SIN(X)
      RETURN
      END
      LOGICAL MORE
C     INPUT THE LIMITS AND THE ACCURACY
      DATA A,B,EPS/0.,3.14159,0.00001/
C     CALL THE ROUTINE FOR INTEGRATION
      CALL ROBSON(A,B,VALUE,EPS,MORE)
      IF(.NOT.MORE)GOTO 11
      WRITE(*,10)
   10 FORMAT(10X,'MORE ITERATIONS REQUIRED')
   11 WRITE(*,12)VALUE,EPS
```

```
C     MODIFY F12.5,E20.4 IF NECESSARY
  12  FORMAT(10X,'VALUE OF INTEGRAL=', F12.5, /
     +10X,' TO AN ACCURACY',E20.4)
      WRITE(*,13)A,B
  13  FORMAT(10X,'THE LIMITS ARE',2F10.4)
      STOP
      END
```

```
INTERMEDIATE VALUE NO 1=2.0000160
INTERMEDIATE VALUE NO 2=2.0000000
INTERMEDIATE VALUE NO 3=2.0000000
VALUE OF INTEGRAL= 2.00000
TO AN ACCURACY          .1000E-04
THE LIMITS ARE        .0000    3.1416
```

7. 列主元Gauss消去法

7.1 功能

本程序采用按列选主元的Gauss（高斯）消去法同时求解系数矩阵相同的m个方程组

$A\boldsymbol{x}_1=\boldsymbol{b}_1$

$A\boldsymbol{x}_2=\boldsymbol{b}_2$

…

$A\boldsymbol{x}_m=\boldsymbol{b}_m$

其中A为$n\cdot n$矩阵，$\boldsymbol{x}_i(i=1,\ 2,\cdots,m)$ 和$\boldsymbol{b}_i$ $(i=1,\ 2,\ \cdots m)$为n个元素的列向量。上列各式可以简写为

$AX=B$

其中$X=[\boldsymbol{x}_1,\boldsymbol{x}_2,\ \cdots,\ \boldsymbol{x}_m]$，$B=[\boldsymbol{b}_1,\ \boldsymbol{b}_2,\ \cdots,\boldsymbol{b}_m]$，它们均是$n\cdot m$矩阵。

7.2 使用说明

1° 子程序调用语句

CALL GAS (N, M, A, B, EPS, L)

2° 虚元说明

输入参数：

N　整变量，方程组的阶数

M　整变量，方程组右端常向量个数

EPS　实变量，消元过程中主元的最小允许值

A　N * N个元素的二维实数组，按列存放方程组的系数矩阵

输入兼输出参数：

B　N * M个元素的二维实数组，开始存放M个方程组的右端项所组成的N * M矩阵；最后存放解向量所组成的N * M矩阵

输出参数：

L　整变量，标志：当矩阵 $A^{(k-1)}$ 的列主元的绝对值小于EPS时，令L＝1，认为（1）无解；否则令L＝0

7.3 例题

求解方程组

$$\begin{pmatrix} 1 & 2 & 1 \\ 3 & 6 & 2 \\ 0 & 3 & 2 \end{pmatrix}\begin{pmatrix} x_{11} \\ x_{12} \\ x_{13} \end{pmatrix}=\begin{pmatrix} 15 \\ 9 \\ 18 \end{pmatrix}$$

$$\begin{pmatrix} 1 & 2 & 1 \\ 3 & 6 & 2 \\ 0 & 3 & 2 \end{pmatrix}\begin{pmatrix} x_{21} \\ x_{22} \\ x_{23} \end{pmatrix}=\begin{pmatrix} 3 \\ -2 \\ 5 \end{pmatrix}$$

取EPS＝10^{-10}.

7.4 程序和计算结果

```
SUBROUTINE GAS(N,M,A,B,EPS,L)
DIMENSION A(N,N),B(N,M)
DO 10 K=1,N
P=0.
DO 20 I=K,N
IF(ABS(A(I,K)).LE.ABS(P)) GOTO 20
```

```
      P=A(I,K)
      IM=I
   20 CONTINUE
      IF(ABS(P)-EPS) 30,30,40
   30 L=1
      RETURN
   40 IF(IM.EQ.K) GOTO 50
      DO 60 J=K,N
      T=A(IM,J)
      A(IM,J)=A(K,J)
   60 A(K,J)=T
      DO 70 J=1,M
      T=B(K,J)
      B(K,J)=B(IM,J)
   70 B(IM,J)=T
   50 P=1./P
      IF(K.EQ.N) GOTO 90
      DO 80 J=K,N-1
      A(K,J+1)=A(K,J+1)*P
      DO 80 I=K,N-1
   80 A(I+1,J+1)=A(I+1,J+1)-A(I+1,K)*A(K,J+1)
   90 DO 100 J=1,M
  100 B(K,J)=B(K,J)*P
      IF(K.EQ.N) GOTO 110
      DO 10 I=K+1,N
      DO 10 J=1,M
      B(I,J)=B(I,J)-A(I,K)*B(K,J)
   10 CONTINUE
  110 DO 120 I1=2,N
      I=N-I1+1
      DO 120 J=1,N-1
      DO 120 K=1,M
  120 B(I,K)=B(I,K)-A(I,J+1)*B(J+1,K)
      L=0
      RETURN
      END

      DIMENSION A(3,3),B(3,2)
      DATA A/1.,3.,0.,2.,6.,3.,1.,2.,2./,
     +B/15.,9.,18.,3.,-2.,5./
      WRITE(*,70)
   70 FORMAT (5X,18(1H),1HA,22(X),1HB)
      DO 50 I=1,3
```

```
      WRITE(*,10) (A(I,J),J=1,3),(B(I,K),K=1,2)
50    CONTINUE
      WRITE(*,20)
10    FORMAT(5X,3F10.5,1H:,3X,2F7.3)
20    FORMAT(8X,44(1H=))
      CALL GAS(3,2,A,B,0.1E-9,L)
      WRITE(*,80)
80    FORMAT(42X,2HX1,5X,2HX2)
      DO 60 I=1,3
      WRITE(*,10) (A(I,J),J=1,3),(B(I,K),K=1,2)
60    CONTINUE
      WRITE(*,20)
      WRITE(*,30) L
30    FORMAT(8X,2HL=,I1)
      STOP
      END
```

```
                  A                                B
1.00000      2.00000      1.00000 :     15.000      3.000
3.00000      6.00000      2.00000 :      9.000     -2.000
 .00000      3.00000      2.00000 :     18.000      6.000
=========================================================
                                          X1         X2
3.00000      2.00000       .66667 :     15.000      3.333
1.00000      3.00000       .66667 :    -18.000     -5.667
 .00000       .00000       .33333 :     36.000     11.000
=========================================================
L=0
```

8. 大型对称正定变带宽方程组的解法

8.1 功能

本程序采用Cholesky(乔莱斯基)法求解大型线性代数方程组

$AX=B$

其中A为$n*n$稀疏对称正定矩阵，X和B均为$n*m$矩阵。对A采用变带宽存储，即每行只存从第一个非零元素到对角线为止的元素(详见7.20)。

8.2 使用说明

1° 子程序调用语句

CALL BAND (N, M, NP, A, B, ID)

2° 虚元说明

输入参数：

N 整变量，方程组的阶数

M 整变量，方程组右端常向量的个数

NP 整变量，系数矩阵A的压缩存放的元素个数

A NP个元素的一维实数组，存放系数矩阵的压缩存储元素

ID N个元素的一维整数组，存放系数矩阵A的各个对角线元素在压缩存贮的一维数组中位置的下标

输入兼输出参数：

B N＊M个元素的二维实数组．开始时存放方程组右端的m个n维常向量；最后存放m个解向量

3° 附注

当系数矩阵奇异或接近奇异时，停机并打印STOP 4444(说明矩阵输入有错)。

8.3 例题

求解线性方程组

$AX=B$

其中A是对称正定矩阵

$$A=\begin{pmatrix} 3.2 & & & & & \\ 0.5 & 7.2 & & \text{对称部分} & & \\ -2.1 & 0 & 8.7 & & & \\ 0 & 0 & 0 & 4.3 & & \\ 0 & 0 & 0 & 2.1 & 9.4 & \\ 0 & 0 & -0.5 & 1.8 & 0 & 5.2 \end{pmatrix},\quad B=\begin{pmatrix} 1.6 \\ 7.7 \\ 6.1 \\ 8.2 \\ 11.5 \\ 6.5 \end{pmatrix}$$

对A用压缩存贮方式，由数组A和ID唯一确定

A＝ (3.2, 0.5, 7.2, －2.1, 0.0, 8.7, 4.3, 2.1, 9.4, －0.5, 1.8, 0.0, 5.2)

ID= (1, 3, 6, 7, 9, 13)

数组A和ID的含义见使用说明。在本题中，N = 6，M = 1，NP = 13。

8.4 程序和计算结果

```
      SUBROUTINE BAND(N,M,NP,A,B,ID)
      DIMENSION A(NP),B(N,M),ID(N)
      DO 10 I=1,N
      I0=ID(I)-I
      IF(I.EQ.1) GOTO 40
      MI=ID(I-1)-I0+1
      DO 20 J=MI,I
      J0=ID(J)-J
      MJ=1
      IF(J.GT.1) MJ=ID(J-1)-J0+1
      MIJ=MI
      IF(MJ.GT.MI) MIJ=MJ
      IJ=I0+J
      JM1=J-1
      DO 30 K=MIJ,JM1
      IF(MIJ.GT.JM1) GOTO 30
      IK=I0+K
      KK=ID(K)
      JK=J0+K
      A(IJ)=A(IJ)-A(IK)*A(KK)*A(JK)
30    CONTINUE
      IF (J.EQ.I) GOTO 40
      JJ=ID(J)
      A(IJ)=A(IJ)/A(JJ)
      DO 20 K=1,M
20    B(I,K)=B(I,K) -A(IJ)*A(JJ)*B(J,K)
40    II=I0+I
      IF(A(II).LT.1E-20) STOP 4444
      DO 10 K=1,M
10    B(I,K)=B(I,K)/A(II)
      DO 50 L=2,N
      I=N-L+2
      I0=ID(I)-I
      MI=ID(I-1)-I0+1
      IM1=I-1
      DO 50 J=MI, IM1
      IF (MI.GT.IM1) GOTO 50
```

```
      IJ=I0+J
      DO 70 K=1, M
   70 B(J,K)=B(J,K)-A(IJ)*B(I,K)
   50 CONTINUE
      RETURN
      END
```

```
      DIMENSION A(13),B(6,1),ID(6)
      DATA A/3.2,0.5,7.2,-2.1,0.,8.7,4.3,2.1,9.4. -0.5,
     +1.8, 0.,5.2/,
     +B/1.6,7.7,6.1,8.2,11.5,6.5/,ID/1,3,6,7,9,13/
      CALL BAND(6,1,13,A,B,ID)
      WRITE(*,20)B
   20 FORMAT(6X,2HX=,6F7.4)
      STOP
      END
```

```
      X= 1.0000 1.0000 1.0000 1.0000 1.0000 1.0000
```

9. 对称带型方程组的解法

9.1 功能

本程序采用Cholesky（乔莱斯基）法（详见6.5）求解带型线性方程组

$AX=B$

其中A为$n*n$对称带型矩阵，即

$a_{ij}=0$ （$|i-j|\geqslant L$）

其中L为半带宽，它是每行第一个非零元素到对角线元素之间元素个数的最大值，存贮方式是将下半带型的左上角添加一些元素使之成为一矩形数组A(N,L)进行存贮（详见7.20），X和B均可为$n*m$矩阵。

9.2 使用说明

$$
\begin{pmatrix}
4 & -1 & 0 & 0 & -1 & & & & & & & & & & & \\
-1 & 4 & -1 & 0 & 0 & -1 & & & & & & & & & & \\
0 & -1 & 4 & -1 & 0 & 0 & -1 & & & & & & & & & \\
0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & & & & & & & & \\
-1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & & & & & & & \\
 & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & & & & & & \\
 & & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & & & & & \\
 & & & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 & & & & \\
 & & & & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 & -1 & & & \\
 & & & & & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & & \\
 & & & & & & -1 & 0 & 0 & -1 & 4 & -1 & 0 & 0 & -1 & \\
 & & & & & & & -1 & 0 & 0 & -1 & 4 & 0 & 0 & 0 & -1 \\
 & & & & & & & & -1 & 0 & 0 & 0 & 4 & -1 & 0 & 0 \\
 & & & & & & & & & -1 & 0 & 0 & -1 & 4 & -1 & 0 \\
 & & & & & & & & & & -1 & 0 & 0 & -1 & 4 & -1 \\
 & & & & & & & & & & & -1 & 0 & 0 & -1 & 4
\end{pmatrix}
\begin{pmatrix}
x_1 \\ x_2 \\ x_3 \\ x_4 \\ x_5 \\ x_6 \\ x_7 \\ x_8 \\ x_9 \\ x_{10} \\ x_{11} \\ x_{12} \\ x_{13} \\ x_{14} \\ x_{15} \\ x_{16}
\end{pmatrix}
=
\begin{pmatrix}
0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04 \\ 0.04
\end{pmatrix}
$$

1° 子程序调用语句

CALL BDS (A, N, IWB, P, M, IE)

2° 虚元说明

输入参数：

A　N*IWB个元素的二维实数组，存放系数矩阵下半带区内的元素，存贮方式见例题

N　整变量，方程组的阶数

IWB　整变量，矩阵A的对角线及下半带区的非零次对角线的数目

M　整变量，方程组右端列向量的个数

输入兼输出参数：

P　N*M个元素的二维实数组。开始存放按列排列的方程组右端M个N维常向量所组成的矩阵；工作结束时存放解向量所组成的矩阵

输出参数：

IE　整变量，标志：IE=0时，表示程序正常结束；当IE=1时，表示系数矩阵奇异

9.3 例题

求解线性方程组（见775页）

即

$Ax=b$

其中A为对称正定、稀疏矩阵，带宽为9，半带宽为5。

9.4 程序和计算结果

```
      SUBROUTINE BDS(A,N,IWB,P,M,IE)
      DIMENSION A(N,IWB),P(N,M)
      DO 15 I=1,N
      IF(I.LE.IWB)GO TO 20
      IT=I-IWB+1
      GO TO 30
   20 IT=1
```

```
30  K=I-1
    IF(I.EQ.1)GO TO 40
    DO 25 L=IT,K
    IL=L+IWB-I
    B=A(I,IL)
    A(I,IL)=B/A(L,IWB)
    DO 35 LL=1,M
35  P(I,LL)=P(I,LL)-A(I,IL)*P(L,LL)
    MI=I+1
    DO 25 J=MI,I
    IJ=J+IWB-I
    JL=L+IWB-J
25  A(I,IJ)=A(I,IJ)-A(J,JL)*B
40  IF(A(I,IWB).EQ.0.)GO TO 100
15  CONTINUE
    DO 45 J=1,N
    IF (J.LE.IWB)GO TO 60
    IT=N-J+IWB
    GO TO 70
60  IT=N
70  I=N-J+1
    DO 55 L=1,M
55  P(I,L)=P(I,L)/A(I,IWB)
    IF(J.EQ.1)GO TO 45
    K=I+1
    DO 65 MJ=K,IT
    IJ=I-MJ+IWB
    DO 65 L=1,M
65  P(I,L)=P(I,L)-P(MJ,L)*A(MJ,IJ)
45  CONTINUE
    IE=0
    GO TO 110
100 IE=1
110 RETURN
    END

    DIMENSION A(16,5),P(16,2)
    DATA A/4*0.,12*-1.,16*0.0,16*0.0,0.0,
   +3*-1.0,0.0,3*-1.0,0.0,3*-1.0,0.0,
   +3*-1.0,16*4./,P/16*0.04,16*0.08/
    WRITE(*,10)((A(I,J),J=1,5),I=1,16)
10  FORMAT(16X,'A'/(4X,5F8.5))
    CALL BDS(A,16,5,P,2,IE)
```

```
      WRITE( * ,20)(P(I,1),I=1,16)
20    FORMAT(16X,'X1'/(4X,5F8.5))
      WRITE( * ,30)(P(I,2),I=1,16)
30    FORMAT(16X,'X2'/(4X,5F8.5))
      WRITE( * ,40)IE
40    FORMAT(4X,'IE',I5)
      STOP
      END
```

```
                 A
   .0000        .0000        .0000         .0000       4.0000
   .0000        .0000        .0000       -1.0000       4.0000
   .0000        .0000        .0000       -1.0000       4.0000
   .0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000         .0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000         .0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000         .0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
 -1.0000        .0000        .0000       -1.0000       4.0000
                 X1
   .03333       .04667       .04667        .03333       .04667
   .06667       .06667       .04667        .04667       .06667
   .06667       .04667       .03333        .04667       .04667
   .03333
                 X2
   .06667       .09333       .09333        .06667       .09333
   .13333       .13333       .09333        .09333       .13333
   .13333       .09333       .06667        .09333       .09333
   .06667
   IE    0
```

10. Guss-Seidel迭代法求大型稀疏线性方程组的解

10.1 功能

本程序用Gauss－Seidel（高斯-赛得尔）迭代法求解线性代数

方程组

$Ax=b$

其中A为$n*n$的大型稀疏矩阵，x和b是n维列向量.对A 采用按行随机存储的方式（见7.20）.

10.2 使用说明

1° 子程序调用语句

```
CALL GS(A,JA,ISTART,N,NP1,IADIM,B,U,ZEAT,
+SR, IADAPT, ITMAX, NUMITS, D)
```

2° 虚元和工作单元说明

输入参数：

A　　　一维实数组，存放系数矩阵的非零元素

JA　　　一维整数组，存放非零系数在原始系数矩阵中的列号

ISTART 一维整数组，存放系数矩阵每一行第一个非零元素在A和JA中的位置，最后一个元素是IADIM+1

N　　　整变量，方程组的阶数

NP1　　整变量，存放N+1

IADIM　整变量，数组A和数组JA的维数

B　　　N个元素的一维实数组，方程组的右端项

ZETA　　实变量，迭代终止标准为：相对误差 ERROR<ZETA

SR　　　实变量，迭代矩阵谱半径(当IADAPT=0 时必须输入)

IADAPT 整变量，=0表示对SR不作自适应估计；=1表示对SR作自适应估计

ITMAX　整变量，最大迭代次数

输入兼输出参数：

U　　　N个元素的一维数组，开始存放解的初始值，最后存放解向量

输出参数：

NUMITS 整变量，迭代执行次数

工作单元：

D　　　N个元素的一维数组，存放对角线元素

PSPREV　实变量，存放前一迭代步的伪-残余向量范数

PSNORM　实变量，存放本迭代步的伪-残余向量范数

UNORM　实变量，本迭代步的解向量范数

TEST　实变量，迭代终止试验中用的单元

TEMP　实变量，新解分量的中间存储单元

10.3 例题

同附录9中的例题．要求迭代精度为2×10^{-6}．

10.4 程序和计算结果

```
      SUBROUTINE GS(A,JA,ISTART,N,NP1,IADIM,B,U, ZEAT,
     +SR,IADAPT,ITMAX,NUMITS,D)
      REAL A(IADIM),B(N),U(N),ZETA,SR
      INTEGER JA(IADIM),ISTART(NP1),N,NP1,IADIM,IADAPT,
     +NUMITS,ITMAX
      REAL PSPREV,PSNORM,D(N),UNORM,SUM,TEST,TEMP
      PSPREV=0.
      DO 10 I=1,N
      SUM=B(I)
      DO 20 J=ISTART(I),ISTART(I+1)-1
      IF(JA(J).EQ.I)THEN
      D(I)=A(J)
      ELSE
      SUM=SUM-A(J)*U(JA(J))
      ENDIF
   20 CONTINUE
      TEMP=SUM/D(I)
      PSPREV=PSPREV+(TEMP-U(I))**2
      U(I)=TEMP
   10 CONTINUE
      PSPREV=SQRT(PSPREV)
      NUMITS=1
   40 NUMITS=NUMITS+1
      IF(NUMITS.GT.ITMAX)RETURN
      PSNORM=0.
      UNORM=0.
      DO 50 I=1,N
```

```
      SUM=B(I)
      DO 60 J=ISTART(I),ISTART(I+1)-1
      IF(JA(J).NE.I)SUM=SUM-A(J)*U(JA(J))
60    CONTINUE
      TEMP=SUM/D(I)
      PSNORM=PSNORM+(TEMP-U(I))**2
      UNORM=UNORM+TEMP**2
      U(I)=TEMP
50    CONTINUE
      PSNORM=SQRT(PSNORM)
      UNORM=SQRT(UNORM)
      IF(IADAPT.EQ.1)SR=PSNORM/PSPREV
      TEST=SR*PSNORM/(1.-SR)/UNORM
      IF(ABS(TEST).LT.ZETA)RETURN
      PSPREV=PSNORM
      GO TO 40
      END
```

```
      DIMENSION A(64), JA(64), ISTART(17), B(16), U(16),D(16)
      DATA A/4, -1, -1, -1, 4, -1, -1, -1, 4, -1, -1, -1,
     +4, -1, -1, 4, -1, -1, -1, -1, 4, -1, -1, -1, -1, 4, -1,
     +-1, -1, -1, 4, -1, -1, 4, -1, -1, -1, -1, 4, -1, -1,
     +-1, -1, 4, -1, -1, -1, -1, 4, -1, -1, 4, -1, -1, -1, 4,
     +-1, -1, -1, 4, -1, -1, -1, 4/, JA/1, 2, 5, 1, 2, 3, 6, 2,
     +3, 4, 7, 3, 4, 8, 1, 5, 6, 9, 2, 5, 6, 7, 10, 3, 6, 7, 8, 11, 4,
     +7, 8, 12, 5, 9, 10, 13, 6, 9, 10, 11, 14, 7, 10, 11, 12, 15, 8,
     +11, 12,16,9, 13, 14, 10, 13, 14, 15, 11, 14, 15, 16, 12, 15, 16/,
     +ISTART/1, 4, 8, 12, 15, 19, 24, 29, 33, 37, 42, 47, 51, 54, 58,
     +62, 65/, B/16*0.04/, U/16*0./, IADAPT/1/
      CALL GS (A, JA, ISTART, 16, 17, 64, B, U, 0.000002, SR,1,
     +ITMAX, NUMITS, D)
      WRITE (*, 1) NUMITS
1     FORMAT('NUMITS=', I2)
      WRITE (*, 2) U
2     FORMAT (6X, 4F9.7)
      WRITE (*, 3) SR
3     FORMAT (6X,' SR=', F9.7)
      STOP
      END
```

```
NUMITS=34
```

.0333333	.0466667	.0466667	.0333333
.0466667	.0666667	.0666667	.0466667
.0466667	.0666667	.0666667	.0466667
.0333333	.0466667	.0466667	.0333333

SR=.654508

11. SOR迭代法求大型稀疏线性方程组的解

11.1 功能

本程序用自适应SOR迭代法求解线性代数方程组

$Ax=b$

其中A为$n*n$矩阵，x和b是n维列向量.

选取最优松弛因子ω_{opt}的自适应方法（详见7.8.9），对A采用按行随机存储的方式（见7.20）.

11.2 使用说明

1° 子程序调用语句

CALL SOR (A, JA, ISTART, N, NP1, IADIM, B,
+U,ZETA,SR,OMEGA,IADAPT,ITMAX,NUMITS,D, H)

2° 虚元和工作单元说明

输入参数：

A	一维实数组，存放系数矩阵的非零元素
JA	一维整数组，存放非零系数在原始系数矩阵中的列号
ISTART	一维整数组，存放系数矩阵每一行第一个非零元素在A和JA中的位置.最后一个元素是IADIM+1
N	整变量，方程组的阶数
NP1	整变量，存放N+1
IADIM	整变量，数组A和数组JA的维数
B	N个元素的一维实数组，方程组的右端项
U	N个元素的一维实数组，开始存放解的初始值，最后存放解向量

ZETA　实变量，迭代终止标准为：相对误差ERROR<ZETA
SR　实变量，迭代矩阵谱半径（如果IADAPT＝0，则必须输入）
OMEGA　实变量，超松弛参数（如果IADAPT＝0，则必须输入）
IADAPT　整变量，＝0表示对SR和OMEGA不作自适应估计；＝1表示自适应
ITMAX　整变量，最大迭代次数

输出参数：

U　解向量，N个元素的一维数组
NUMITS　整变量，迭代执行次数

工作单元：

D　N个元素的一维数组，存放对角线元素
PSPREV　实变量，存放前一迭代步的残余向量范数
PSNORM　实变量，本迭代步的残余向量范数
UNORM　实变量，本迭代步的解向量范数
TEST　实变量，迭代终止试验中用的单元
H　实变量，终止试验中所用的单元
TEMP　实变量，新的解分量的中间存贮单元
SRJ　实变量，Jacobi矩阵谱半径
T　一维实数组，每个OMEGA估值的上界
IEST　整变量，不相同的OMEGA估值的个数
NEWITS　整变量，用于新估值的迭代次数
CHANGE　实变量，检验SR是否为合适的估值

11.3　例题

同附录9中的例题，要求迭代精度为10^{-5}。

11.4　程序和计算结果

```
SUBROUTINE SOR(A,JA,ISTART,N,NP1,IADIM,B,U,ZETA,
```

```
     +SR, OMEGA, IADAPT, ITMAX, NUMITS, D, H)
      REAL A (IADIM), B(N), U(N), ZETA, SR, OMEGA
      INTEGER JA(IADIM),ISTART(NP1),N,NP1,IADIM,IADAPT,
     +NUMITS, ITMAX
      REAL  PSPREV,PSNORM,D(N), UNORM, SUM, TEST, TEMP,
     +H,SRJ,T(9), CHANGE
      INTEGER IEST, NEWITS
      DATA T/1.5, 1.8, 1.85, 1.9, 1.94, 1.96, 1.975, 1.985,1.992/
      IF (IADAPT. EQ. 1) THEN
      IEST=1
      OMEGA=MIN (OMEGA, T(1))
      ENDIF
      PSPREV=0.
      DO 10 I=1, N
      SUM=B(I)
      DO 20 J=ISTART(I), ISTART (I+1)-1
      IF (JA(J). EQ. I) THEN
      D(I)=A(J)
      ELSE
      SUM=SUM-A(J) * U(JA(J))
      ENDIF
20    CONTINUE
      TEMP=OMEGA * SUM/D(I)+(1.0-OMEGA) * U(I)
      PSPREV=PSPREV+(TEMP-U(I)) * * 2
      U(I)=TEMP
10    CONTINUE
      PSPREV=SQRT (PSPREV)
      NUMITS=1
      NEWITS=1
40    NUMITS=NUMITS+1
      IF (NUMITS. GT. ITMAX) RETURN
      NEWITS=NEWITS+1
      PSNORM=0.
      UNORM=0.
      DO 50 I=1, N
      SUM=B(I)
      DO 60 J=ISTART(I), ISTART(I+1)-1
      IF (JA(J). NE. I) SUM=SUM-A(J) * U(JA(J))
60    CONTINUE
      TEMP=OMEGA * SUM/D(I)+(1.0-OMEGA) * U(I)
      PSNORM=PSNORM+(TEMP-U(I)) * * 2
      UNORM=UNORM+TEMP * * 2
      U(I)=TEMP
```

```
50  CONTINUE
    PSNORM=SQRT (PSNORM)
    UNORM=SQRT (UNORM)
    IF (IADAPT. EQ. 0) THEN
    TEST=PSNORM/(1.-SR)/UNORM
    IF (ABS (TEST). LT. ZETA) RETURN
    GO TO 80
    ENDIF
    IF (NEWITS. LT. 2) GO TO 80
    IF (ABS (SR-PSNORM/PSPREV). GT. 0. 01) THEN
    CHANGE=0.
    ELSE
    CHANGE=1.
    ENDIF
    SR=PSNORM/PSPREV
    IF (SR. GE. 1.0) GO TO 80
    IF (SR. LT. (OMEGA-1.0)) THEN
    H=OMEGA-1.0
    ELSE
    H=SR
    ENDIF
    TEST=PSNORM/(1.-H)/UNORM
    IF (ABS(TEST). LT. ZETA) RETURN
    IF(NEWITS.LT.5.OR.SR.LT.(OMEGA-1.)**0.75.OR.CHANGE
   +. EQ. 0.) GO TO 80
    IEST=IEST+1
    NEWITS=0
    SRJ=(SR+OMEGA-1.0)/OMEGA/SQRT (SR)
    OMEGA=2.0/(1.0+SQRT (1.0-SRJ**2))
    IF (IEST. LE. 9) THEN
    OMEGA=MIN (OMEGA, T (IEST))
    ELSE
    OMEGA=MIN (OMEGA, T(9))
    ENDIF
80  PSPREV=PSNORM
    GO TO 40
    END
```

```
    DIMENSION A(64), JA(64), ISTART(17), B(16), U(16), D(16)
    DATA A/4, -1, -1, -1, 4, -1, -1, -1, 4, -1, -1, -1,
   +4, -1, -1, 4, -1, -1, -1, -1, 4, -1, -1, -1, -1, 4,-1,
```

```
+-1, -1, -1, 4, -1, -1, 4, -1, -1, -1, -1, 4 , -1 , -1 ,
+-1, -1, 4, -1, -1, -1, -1, 4, -1, -1, 4, -1, -1,-1, 4,
+-1, -1, -1, 4, -1, -1, -1, 4/, JA/1,2, 5, 1, 2, 3, 6, 2, 3,
+4, 7, 3, 4, 8, 1, 5, 6, 9, 2, 5, 6, 7, 10, 3, 6, 7, 8, 11, 4, 7,
+8, 12, 5, 9, 10, 13, 6, 9, 10, 11, 14, 7, 10, 11, 12, 15, 8, 11,
+12, 16, 9, 13, 14, 10, 13, 14, 15, 11, 14, 15, 16, 12, 15, 16/,
+ISTART/1, 4, 8, 12, 15, 19, 24, 29, 33, 37, 42, 47, 51, 54,
+58, 62, 65/, B/16*0.04/, U/16*0./, OMEGA/1./
 CALL SOR (A, JA, ISTART, 16, 17, 64, B, U, 0.00001, SR,
+OMEGA, 1, 100, NUMITS, D, H)
 WRITE(*,1) OMEGA
1 FORMAT (6X,' OMEGA=',F9.6)
 WRITE (*, 2) NUMITS
2 FORMAT (6X,' NUMITS=', I3)
 WRITE (*, 3) U
3 FORMAT (6X, 4F9.7)
 WRITE (*, 4) H
4 FORMAT (6X,' H=',F9.7)
 STOP
 END
```

```
OMEGA=1.282128
NUMITS=15
  .0333335    .0466668    .0466668    .0333334
  .0466668    .0666668    .0666667    .0466667
  .0466668    .0666667    .0666667    .0466667
  .0333334    .0466667    .0466667    .0333333
H=.3709074
```

12. Чебышев加速法加速Jacobi迭代求解大型稀疏线性方程组

12.1 功能

本程序用Чебышев（切比雪夫）多项式加速Jacobi（雅可比）迭代法求解线性方程组

$Ax=b$

其中A为$n*n$大型稀疏矩阵，x和b是n维列向量。对A采用按行随机存储的方式（见7.20）。

12.2 使用说明

1° 子程序调用语句

```
 CALL  JSI(A,JA,ISTART,N, NP1, IADIM,B,U,ZETA,
+SRJ, SRNJ, ICASE, IADAPT, IT MAX, NUMITS, V,
+W,D)
```

2°虚元和工作单元说明

输入参数：

A　　一维实数组，存放系数矩阵的非零元素

JA　　一维整数组，存放非零系数在原始系数矩阵中的列号

ISTART　一维整数组，存放系数矩阵每一行第一个非零元素在A和JA中的位置。最后一个元素是IADIM+1

N　　整变量，方程组的阶数

NP1　　整变量，存放N+1

IADIM　整变量，数组A和数组JA的维数

B　　N个元素的一维实数组，方程组的右端项

ZETA　实变量，迭代终止标准为：相对误差ERROR<ZETA

SRJ　　实变量，Jacobi迭代矩阵的谱半径（必须输入）

SRNJ　实变量，最小特征值（大多数为负）

ICASE　整变量，=0表示不改变SRNJ的值；=1表示SRNJ=-SRJ

IADAPT　整变量，=0表示对SRJ，SRNJ不作自适应估计；=1表示自适应

ITMAX　整变量，最大迭代次数

输入兼输出参数：

U　　N个元素的一维实数组，开始存放解的初始值，最后

存放解向量

输出参数：

NUMITS　整变量，迭代执行次数

工作单元：

V	N个元素的一维实数组，临时工作单元。
W	N个元素的一维实数组，存放前一次的迭代值
D	N个元素的一维实数组，存放矩阵的对角线元素
PSORIG	实变量，前一迭代步的伪-残余范数
PSNORM	实变量，本迭代步的伪-残余范数
UNORM	实变量，本迭代步的解向量范数
TEST	实变量，迭代终止试验中用的单元
GAMMA	实变量，参数
SIGMA	实变量，参数
R	实变量，参数
OMEGA	实变量，加速参数
TEMP	实变量，临时工作单元
QA	实变量，实际的残商
QT	实变量，理论的残商
NEWITS	实变量，采用新估值的迭代次数

3°　附注

本程序开始时使SRJ=0，SRNJ=-1，当迭代二次到六次以后，程序自适应求得较好的SRJ，程序自动使SRNJ=-SRJ。从下面的计算结果可以看出，迭代六次后SRJ就不再改变了。如果不需要打印每一步的计算结果，可以将子程序JSI中的打印语句去掉，仅输出最后收敛解。

12.3　例题

同附录9中的例题，要求迭代精度为10^{-4}。

12.4 程序和计算结果

```
      SUBROUTINE JSI (A,JA,ISTART,N,NP1,IADIM,B,U,ZETA,
     +SRJ, SRNJ, ICASE, IADAPT, ITMAX, NUMITS, V, W, D)
      REAL A (IADIM), B(N), U(N), ZETA, SRJ, SRNJ
      INTEGER JA(IADIM),ISTART(NP1),N,NP1,IADIM, ICASE,
     +IADAPT, NUMITS, ITMAX
      REAL PSORIG,PSNORM,V(N),W(N),D(N),UNORM,SUM,TEST,
     +TEMP, GAMMA, SIGMA, R, OMEGA, QA, QT
      DO 10 I=1, N
      DO 20 J=ISTART(I), ISTART (I+1)-1
      IF (JA (J). EQ. I) THEN
      D(I)=A(J)
      GOTO 10
      ENDIF
   20 CONTINUE
   10 CONTINUE
      NUMITS=0
   25 GAMMA=2.0/(2.0-SRJ-SRNJ)
      SIGMA= (SRJ-SRNJ)/(2.0-SRJ-SRNJ)
      R=SQRT (1.0-SIGMA * * 2)
      R=(1.0-R)/(1.0+R)
      NEWITS=0
   30 NUMITS=NUMITS+1
      IF (NUMITS. GT. ITMAX) RETURN
      NEWITS=NEWITS+1
      PSNORM=0.
      UNORM=0.
      IF (NEWITS. EQ. 1) GOTO 40
      IF (NEWITS. EQ. 2) THEN
      OMEGA=1.0/(1.0-0.5 * SIGMA * * 2)
      ELSE
      OMEGA=1.0/(1.0-0.25 * OMEGA * SIGMA * * 2)
      ENDIF
   40 DO 50 I=1, N
      SUM=B(I)
      DO 60 J=ISTART(I), ISTART (I+1)-1
      IF (JA(J). NE. I) SUM=SUM-A(J) * U (JA(J))
   60 CONTINUE
      TEMP=SUM/D(I)-U(I)
      PSNORM=PSNORM+TEMP * * 2
      IF(NEWITS.EQ. 1) THEN
      V(I)=GAMMA * TEMP+U(I)
```

```
      ELSE
      V(I)=OMEGA*(GAMMA*TEMP+U(I))+(1.0-OMEGA)*W(I)
      ENDIF
      UNORM=UNORM+V(I)**2
 50   CONTINUE
      UNORM=SQRT (UNORM)
      PSNORM=SQRT (PSNORM)
      TEST=PSNORM/(1.0-SRJ)/UNORM
      IF (ABS(TEST). LT. ZETA) RETURN
      WRITE (*, 200) NUMITS
200   FORMAT (6X,' NUMITS', I3)
      WRITE (*, 210)  (U(I), I=1, N)
210   FORMAT (6X, 4F9.7)
      WRITE (*, 220) SRJ
220   FORMAT (6X,' SRJ=', F9.6)
      IF (NEWITS. EQ. 1) PSORIG=PSNORM
      DO 70 I=1, N
      W(I)=U(I)
      U(I)=V(I)
 70   CONTINUE
      IF(IADAPT. EQ. 0. OR. NEWITS. EQ. 1) GOTO 30
      QA=PSNORM/PSORIG
      TEMP=R**(NEWITS-1)
      QT=2.0*SQRT (TEMP)/(1.0+TEMP)
      IF(QA. LT.QT**0.75) GOTO 30
      TEMP=0.5*(1.0+TEMP)*(QA+SQRT (QA*QA-QT*QT))
      TEMP=TEMP**(1.0/(FLOAT (NEWITS)-1.0))
      TEMP=(TEMP+R/TEMP)/(1.0+R)
      SRJ=0.5*(SRJ+SRNJ+TEMP*(2.0-SRJ-SRNJ))
      IF(ICASE. EQ. 1)SRNJ=-SRJ
      GOTO 25
      END
```

```
      DIMENSION A(64), JA(64), ISTART(17), B(16),
     +U(16), D(16), W(16), V(16)
      DATA A/4., -1., -1., -1., 4., -1., -1., -1., 4.,
     +-1., -1., -1., 4., -1., -1., 4., -1., -1., -1., -1.,
     +4., -1., -1., -1., -1., 4., -1., -1., -1., -1., 4.,
     +-1., -1., 4., -1., -1., -1., -1., 4., -1., -1., -1.,
     +-1., 4., -1., -1., -1., -1., 4., -1., -1., 4., -1.,
     +-1., -1., 4., -1., -1., -1., 4., -1., -1., -1., 4./
```

```
      DATA JA/1, 2, 5, 1, 2, 3, 6, 2, 3, 4, 7, 3, 4, 8, 1, 5,
     +6, 9, 2, 5, 6, 7, 10, 3, 6, 7, 8, 11, 4, 7, 8, 12, 5, 9,
     +10, 13, 6, 9, 10, 11, 14, 7, 10, 11, 12, 15, 8, 11,
     +12, 16, 9, 13, 14, 10, 13, 14, 15, 11, 14, 15, 16, 12,
     +15, 16/
      DATA ISTART/1, 4, 8, 12, 15, 19, 24, 29, 33, 37, 42,
     +47, 51, 54, 58, 62, 65/
      DATA B/16*0.04/, U/16*0.0/, SRJ/0.0/,
     +SRNJ/-1.0/
      CALL JSI (A, JA, ISTART, 16, 17, 64, B, U, 0.0001,
     +SRJ, SRNJ, 0, 1, 100, NUMITS, V, W, D)
      WRITE (*, 10) NUMITS
   10 FORMAT (6X,' NUMITS', I3)
      WRITE (*, 20) U
   20 FORMAT (6X, 4F9.7)
      WRITE (*, 30) SRJ
   30 FORMAT (6X,' SRJ=', F9.6)
      STOP
      END
      NUMITS 1
        .0000000  .0000000  .0000000  .0000000
        .0000000  .0000000  .0000000  .0000000
        .0000000  .0000000  .0000000  0000000
        .0000000  .0000000  .0000000  .0000000
      SRJ=  .000000
      NUMITS 2
        .0066667  .0066667  .0066667  .0066667
        .0066667  .0066667  .0066667  .0066667
        .0066667  .0066667  .0066667  .0066667
        .0066667  .0066667  .0066667  .0066667
      SRJ=  .000000
      NUMITS 3
        .0117647  .0129412  .0129412  .0117647
        .0129412  .0141176  .0141176  .0129412
        .0129412  .0141176  .0141176  .0129412
        .0117647  .0129412  .0129412  .0117647
      SRJ= .762438
      NUMITS 4
        .0159710  .0189877  .0189877  .0159710
        .0189877  .0225302  .0225302  .0189877
        .0189877  .0225302  .0225302  .0189877
        .0159710  .0189877  .0189877  .0159710
      SRJ= .762438
```

NUMITS 5

.0224274	.0286840	.0286840	.0224274
.0286840	.0369761	.0369761	.0286840
.0286840	.0369761	.0369761	.0286840
.0224274	.0286840	.0286840	.0224274

SRJ= .762438

NUMITS 6

.0265080	.0353456	.0353456	.0265080
.0353456	.0479168	.0479168	.0353456
.0353456	.0479168	.0479168	.0353456
.0265080	.0353456	.0353456	.0265080

SRJ= .762438

NUMITS 7

.0288303	.0393552	.0393552	.0288303
.0393552	.0548041	.0548041	.0393552
.0393552	.0548041	.0548041	.0393552
.0288303	.0393552	.0393552	.0288303

SRJ= .808694

NUMITS 8

.0296036	.0406259	.0406259	.0296036
.0406259	.0568810	.0568810	.0406259
.0406259	.0568810	.0568810	.0406259
.0296036	.0406259	.0406259	.0296036

SRJ= .808694

NUMITS 9

.0309850	.0428766	.0428766	.0309850
.0428766	.0605458	.0605458	.0428766
.0428766	.0605458	.0605458	.0428766
.0309850	.0428766	.0428766	.0309850

SRJ= .808694

NUMITS 10

.0320239	.0445529	.0445529	.0320239
.0445529	.0632563	.0632563	.0445529
.0445529	.0632563	.0632563	.0445529
.0320239	.0445529	.0445529	.0320239

SRJ= .808694

NUMITS 11

.0326328	.0455329	.0455329	.0326328
.0455329	.0648324	.0648324	.0455329
.0455329	.0648324	.0648324	.0455329
.0326328	.0455329	.0455329	.0326328

SRJ= .808694

NUMITS 12

.0329622	.0460649	.0460649	.0329622
.0460649	.0656901	.0656901	.0460649
.0460649	.0656901	.0656901	.0460649
.0329622	.0460649	.0460649	.0329622

SRJ= .808694

NUMITS 13

.0331364	.0463474	.0463474	.0331364
.0463474	.0661491	.0661491	.0463474
.0463474	.0661491	.0661491	.0463474
.0331364	.0463474	.0463474	.0331364

SRJ= .808694

NUMITS 14

.0332285	.0464972	.0464972	.0332285
.0464972	.0663928	.0663928	.0464972
.0464972	.0663928	.0663928	.0464972
.0332285	.0464972	.0464972	.0332285

SRJ= .808694

NUMITS 15

.0332776	.0465767	.0465767	.0332776
.0465767	.0665215	.0665215	.0465767
.0465767	.0665215	.0665215	.0465767
.0332776	.0465767	.0465767	.0332776

SRJ= .808694

NUMITS 16

.0333038	.0466190	.0466190	.0333038
.0466190	.0665896	.0665896	.0466190
.0466190	.0665896	.0665896	.0466190
.0333038	.0466190	.0466190	.0333038

SRJ= .808694

NUMITS 17

.0333177	.0466414	.0466414	.0333177
.0466414	.0666257	.0666257	.0466414
.0466414	.0666257	.0666257	.0466414
.0333177	.0466414	.0466414	0333177

SRJ= .808694

NUMITS 18

.0333251	.0466532	.0466532	.0333251
.0466532	.0666449	.0666449	.0466532
.0466532	.0666449	.0666449	.0466532
.0333251	.0466532	.0466532	.0333251

SRJ= .808694

NUMITS 19

.0333289	.0466595	.0466595	.0333289

.0466595	.0666551	.0666551	.0466595
.0466595	.0666551	.0666551	.0466595
.0333289	.0466595	.0466595	.0333289

SRJ = .808694
NUMITS 20

.0333310	.0466629	.0466629	.0333310
.0466629	.0666605	.0666605	.0466629
.0466629	.0666605	.0666605	.0466629
.0333310	.0466629	.0466629	.0333310

SRJ = .808694

13. 共轭斜量加速Jacobi迭代法求解大型稀疏线性方程组

13.1 功能

本程序用共轭斜量（Conjucate gradient）加速Jacobi（雅可比）迭代法求解线性代数方程组

$Ax=b$

其中A为$n*n$的大型稀疏矩阵，x和b是n维列向量。对A采用按行随机存储的方式（见7.20）。

13.2 使用说明

1° 子程序调用语句

```
CALL JCG(A, JA,ISTART,N,NP1, IADIM,B,U,ZETA,
+SRJ, IADAPT,ITMAX,NUMITS,W,D,PSU,PSV, PSW)
```

2° 虚元和工作单元说明

输入参数：

A　一维实数组，存放系数矩阵的非零元素

JA　一维整数组，存放非零系数在原始系数矩阵中的列号

ISTART　一维整数组，存放系数矩阵每一行第一个非零元素在A和JA中的位置。最后一个元素是IADIM+1

N　整变量，方程组的阶数

NP1　　整变量，存放N+1

IADIM　整变量，数组A和数组JA的维数

B　　N个元素的一维实数组，方程组的右端项

ZETA　实变量，迭代终止标准为：相对误差 ERROR<ZETA

SRJ　实变量，Jacobi迭代矩阵的谱半径（必须输入）

IADAPT　整变量，=0表示对SRJ不作自适应估计；=1表示自适应

ITMAX　整变量，最大迭代次数

输入兼输出参数：

U　　N个元素的一维实数组，开始存放解的初始值，最后存放解向量

输出参数：

NUMITS　整变量，迭代执行次数

工作单元：

W　　N个元素的一维实数组，存放前一次的迭代值

D　　N个元素的一维实数组，存放对角线元素

PSNORM　实变量，本迭代步的伪-残余范数

UNORM　实变量，本迭代步的解向量范数

TEST　实变量，迭代终止试验中用的单元

PSU　N个元素的一维实数组，存放本迭代步的伪-残余向量

PSV　N个元素的一维实数组，存放新的伪-残余向量

PSW　N个元素的一维实数组，存放前一次迭代的伪-残余向量

PSUPSV　实变量，存放内积

PSUPSU　实变量，存放内积

PSUPSW　实变量，存放内积

GAMMAU　实变量，参数

GAMMAW　实变量，参数

OMEGAU	实变量，本迭代步的加速参数
OMEGAW	实变量，前一迭代步的加速参数
ALPHA	200个元素的一维实数组，存放三对角矩阵的对角元素
BETA	200个元素的一维实数组，存放非对角线元素的平方
CHANGE	实变量，当SRJ足够精确时，它等于零
TEMP	实变量，中间存储单元

3° 附注

本程序适用于$n\leq200$的方程组，当$n>200$时，则需要修改子程序JCG和子程序NEWTON中常界数组ALPHA和BETA的界

13.3 例题

同附录9中的例题，要求迭代精度为10^{-5}。

13.4 程序和计算结果

```
      SUBROUTINE JCG(A,JA,ISTART,N,NP1,IADIM,B,U, ZETA,
     +SRJ, IADAPT, ITMAX, NUMITS, W, D, PSU, PSV, PSW)
      REAL A (IADIM), B(N), U(N), ZETA, SRJ
      INTEGER JA(IADIM), ISTART(NP1),N,NP1,IADIM, IADAPT,
     +NUMITS, ITMAX
      REAL PSNORM, W(N), D(N), PSU(N), PSV(N), PSW(N),
     +UNORM, SUM,TEST,TEMP,PSUPSV,PSUPSU,PSWPSW,
     +GAMMAU,GAMMAW, OMEGAU, OMEGAW, ALPHA(200),
     +BETA(200)
      INTEGER CHANGE
      PSUPSU=0.0
      DO 10 I=1, N
      SUM=B(I)
      DO 20 J=ISTART(I), ISTART(I+1)-1
      IF (JA(J). EQ. I) THEN
      D(I)=A(J)
      ELSE
      SUM=SUM-A(J)*U(JA(J))
      ENDIF
   20 CONTINUE
```

```
      PSU(I)=SUM/D(I)-U(I)
      PSUPSU=PSUPSU+D(I)*PSU(I)**2
   10 CONTINUE
      CHANGE=1
      NUMITS=0
      GAMMAU=0.0
      GOTO 50
   30 IF (CHANGE. EQ. 0. OR. IADAPT. EQ. 0) GOTO 40
      IF (NUMITS. GT. 2) THEN
      CALL NEWTON(SRJ,CHANGE,ZETA,ALPHA,BETA, NUMITS)
      ELSEIF (NUMITS. EQ. 1) THEN
      SRJ=ALPHA(1)
      ELSE
      SRJ=ALPHA(1)+ALPHA(2)
      TEMP=ALPHA(1)-ALPHA(2)
      SRJ=0.5*(SRJ+SQRT (TEMP**2+4.0*BETA(2)))
      ENDIF
      WRITE (*, 35) NUMITS, SRJ
   35 FORMAT (1X,' NUMITS=', I4/1X,' SRJ=', E12.6/)
   40 TEST=PSNORM/(1.0-SRJ)/UNORM
      IF (ABS (TEST).LT. ZETA) RETURN
   50 NUMITS=NUMITS+1
      IF (NUMITS. GT. ITMAX) RETURN
      PSUPSV=0.0
      DO 60 I=1, N
      SUM=0.0
      DO 70 J=ISTART(I), ISTART(I+1)-1
      IF (JA(J). NE. I) SUM=SUM-A(J)*PSU (JA(J))
   70 CONTINUE
      PSV(I)=SUM/D(I)
      PSUPSV=PSUPSV+PSU(I)*D(I)*PSV(I)
   60 CONTINUE
      GAMMAW=GAMMAU
      GAMMAU=1.0/(1.0-PSUPSV/PSUPSU)
      IF(NUMITS. EQ. 1) THEN
      OMEGAU=1.0
      ELSE
      TEMP=1.0/(1.0-PSUPSU*GAMMAU/(OMEGAU*GAMMAW
     + *PSWPSW))
      OMEGAW=OMEGAU
      OMEGAU=TEMP
      ENDIF
      UNORM=0.0
```

```
      PSNORM=0.0
      PSWPSW=PSUPSU
      PSUPSU=0.0
      IF (NUMITS. EQ. 1) THEN
      DO 80 I=1, N
      TEMP=GAMMAU * PSU(I)+U(I)
      W(I)=U(I)
      U(I)=TEMP
      UNORM=UNORM+TEMP * * 2
      TEMP=GAMMAU * PSV(I)+(1.0-GAMMAU) * PSU(I)
      PSW(I)=PSU(I)
      PSU(I)=TEMP
      TEMP=TEMP * * 2
      PSNORM=PSNORM+TEMP
      PSUPSU=PSUPSU+D(I) * TEMP
80    CONTINUE
      ALPHA(1)=1.0-1.0/GAMMAU
      ELSE
      DO 90 I=1, N
      TEMP=OMEGAU * (GAMMAU * PSU(I)+U(I))+
     +(1.0-OMEGAU) * W(I)
      W(I)=U(I)
      U(I)=TEMP
      UNORM=UNORM+TEMP * * 2
      TEMP=OMEGAU * (GAMMAU * PSV(I)+(1.0-GAMMAU)
     + * PSU(I) )+(1.0-OMEGAU) * PSW(I)
      PSW(I)=PSU(I)
      PSU(I)=TEMP
      TEMP=TEMP * * 2
      PSNORM=PSNORM+TEMP
      PSUPSU=PSUPSU+D(I) * TEMP
90    CONTINUE
      ALPHA (NUMITS) =1.0-1.0/GAMMAU
      BETA (NUMITS)=(OMEGAU-1.0)/(GAMMAW * GAMMAU
     + * OMEGAW * OMEGAU)
      ENDIF
      UNORM=SQRT(UNORM)
      PSNORM=SQRT (PSNORM)
      GOTO 30
      END

      SUBROUTINE NEWTON(SRJ, CHANGE, ZETA, ALPHA,
```

```
     +BETA,NUMITS)
      REAL SRJ, ZETA, ALPHA(200), BETA(200)
     +, X, FX(200), DFX(200), DELTAX
      INTEGER CHANGE, NUMITS
      X=SRJ
100   FX(1)=ALPHA(1)-X
      DFX(1)=-1.0
      FX(2)=FX(1)*(ALPHA(2)-X)-BETA(2)
      DFX(2)=DFX(1)*(ALPHA(2)-X)-FX(1)
      DO 110 I=3, NUMITS
      FX(I)=FX(I-1)*(ALPHA(I)-X)-FX(I-2)*BETA(I)
      DFX(I)=DFX(I-1)*(ALPHA(I)-X)-FX(I-1)-DFX(I-2)
     +*BETA(I)
110   CONTINUE
      DELTAX=FX(NUMITS)/DFX(NUMITS)
      X=X-DELTAX
      IF (ABS (DELTAX). GT. ZETA) GOTO 100
      IF (ABS(X-SRJ). LT. ZETA) CHANGE=0
      SRJ=X
      RETURN
      END

      DIMENSION A(64), JA(64), ISTART(17), B(16),
     +U(16), D(16), W(16), PSU(16), PSV(16), PSW(16)
      DATA A/4., -1., -1., -1., 4., -1., -1., -1., 4.,
     +-1., -1., -1., 4., -1., -1., 4., -1., -1., -1., -1.,
     +4., -1., -1., -1., -1., 4., -1., -1., -1., -1., 4.,
     +-1., -1., 4., -1., -1., -1., -1., 4., -1., -1., -1.,
     +-1., 4., -1., -1., -1., -1., 4., -1., -1., 4., -1.,
     +-1., -1., 4., -1., -1., -1., 4., -1., -1., -1., 4./
      DATA JA/1, 2, 5, 1, 2, 3, 6, 2, 3, 4, 7, 3, 4, 8, 1, 5,
     +6, 9, 2, 5, 6, 7, 10, 3, 6, 7, 8, 11, 4, 7, 8, 12, 5, 9,
     +10, 13, 6, 9, 10, 11, 14, 7, 10, 11, 12, 15, 8, 11,
     +12, 16, 9, 13, 14, 10, 13, 14, 15, 11, 14, 15, 16, 12,
     +15, 16/
      DATA ISTART/1, 4, 8, 12, 15, 19, 24, 29, 33, 37, 42,
     +47, 51, 54, 58, 62, 65/
      DATA B/16*0.04/,U/16*0./,SRJ/0.9/
      CALL JCG (A, JA, ISTART, 16, 17, 64, B, U, 0.00001,
     +SRJ, 1, 100, NUMITS, W, D, PSU, PSV, PSW)
      WRITE (*, 10) NUMITS
10    FORMAT (6X,' NUMITS=', I3)
      WRITE (*, 20) U
```

```
20    FORMAT (6X, 4F9.7)
      WRITE (*, 30) SRJ
30    FORMAT (6X,' SRJ=', F9.6)
      STOP
      END
      NUMITS=      1
      SRJ=.750000E+00
      NUMITS=      2
      SRJ=  .806186E+00
      NUMITS=      3
      SRJ=  .809017E+00
      NUMITS=      3
        .0333333     .0466667     .0466667     .0333333
        .0466667     .0666667     .0666667     .0466667
        .0466667     .0666667     .0666667     .0466667
        .0333333     .0466667     .0466667     .0333333
      SRJ=   .809017
```

14. Jacobi方法求实对称矩阵的特征值和特征向量

14.1 功能

本程序用Jacobi（雅可比）方法求实对称矩阵A的全部特征值和特征向量。

14.2 使用说明

1°子程序调用语句

CALL JACOB(A, V, ERR, N, NX, ITM)

2°虚元说明

输入参数：

N　　整变量，矩阵A的阶数

ERR　实变量，允许误差

输入兼输出参数：

A　　N*N个元素的二维实数组，开始存放矩阵元素，最后在它的对角线上是求得的特征值

输出参数：

V　N*N个元素的二维实数组，它的每一列是一组特征向量

14.3 例题

求矩阵

$$A=\begin{bmatrix} 2 & 3 & 2 \\ 3 & 3 & 6 \\ 2 & 6 & 4 \end{bmatrix}$$

的特征值和特征向量。

14.4 程序和计算结果

```
      SUBROUTINE JACOB (A, V, ERR, N, NX  ITM)
      DIMENSION A (NX, NX), V(NX, NX)
      IT=0
C     PUT A UNIT MATRIX IN ARRAY V
      DO 10 I=1, N
      DO 10 J=1, N
      IF (I-J) 3, 1, 3
    3 V(I, J)=0.
      GOTO 10
    1 V(I, J)=1.
   10 CONTINUE
      FIND LARGEST OFF DIAGONAL COEFFICIENT
   13 T=0.
      M=N-1
      DO 20 I=1, M
      J1=I+1
      DO 20 J=J1, N
      IF (ABS (A(I, J))-T) 20, 20, 2
    2 T=ABS (A(I, J))
      IR=I
      IC=J
   20 CONTINUE
      IF (IT) 5, 4, 5
C     TAKE FIRST LARGEST OFF DIAGONAL COEFFICIENT
C     TIMES ERR AS COMPARISON VALUE FOR ZERO
    4 T1=T*ERR
    5 IF (T-T1) 999, 999, 6
```

```
C      COMPUTE TAN (TA), SIN(S) AND COSINE(C)
C      OF ROTATION ANGLE
   6   PS=A(IR, IR)-A(IC, IC)
       TA=(-PS+SQRT(PS*PS+4*T*T))/(2.*A(IR, IC))
       C=1./SQRT(1.+TA*TA)
       S=C*TA
C      MULTIPLY ROTATION MATRIX TIMES V
C      AND STORE IN V
       DO 50 I=1, N
       P=V(I, IR)
       V(I, IR)=C*P+S*V(I, IC)
  50   V(I, IC)=C*V (I, IC) -S*P
       I=1
 100   IF (I-IR) 7, 200, 7
C      APPLY ORTHOGONAL TRANSFORMATION
C      TO A AND STORE IN A
   7   P=A(I, IR)
       A(I, IR)=C*P+S*A(I, IC)
       A(I, IC)=C*A(I, IC)-S*P
       I=I+1
       GOTO 100
 200   I=IR+1
 300   IF (I-IC) 8, 400, 8
   8   P=A(IR, I)
       A(IR, I)=C*P+S*A(I, IC)
       A(I, IC)=C*A(I, IC) -S*P
       I=I+1
       GOTO 300
 400   I=IC+1
 500   IF(I-N) 9, 9, 600
   9   P=A (IR, I)
       A(IR, I)=C*P+S*A(IC, I)
       A(IC, I)=C*A(IC, I) -S*P
       I=1+I
       GOTO 500
 600   P=A(IR, IR)
       A(IR, IR)=C*C*P+2.*C*S*A(IR,IC)+S*S*A (IC,IC)
       A(IC, IC)=C*C* A(IC, IC)+S*S*P-2.*C*S*A(IR, IC)
       A (IR, IC)=0.
       IT=IT+1
       IF (IT-ITM) 13, 13, 999
 999   RETURN
       END
```

```
      DIMENSION A(3, 3), V(3, 3)
      DATA A/2., 3., 2., 3., 3., 6., 2., 6., 4./
      WRITE (*, 5)
 5    FORMAT (20X, 1HA)
      WRITE (*, 10) A
 10   FORMAT (5X, 3F10.5)
      WRITE (*, 15)
 15   FORMAT (5X, 37(1H=)/6X, 11HEIGENVALUE:, 10X,
     +11HEIGENVECTOR)
      CALL JACOB (A, V, 0.E-5, 3,3,200)
      DO 30 I=1, 3
      WRITE (*, 20) A(I, I), (V(J, I), J=1, 3)
 20   FORMAT (5X, F10.5, 3H:, 3F8.3)
 30   CONTINUE
      STOP
      END
```

```
                    A
     2.00000      3.00000      2.00000
     3.00000      3.00000      6.00000
     2.00000      6.00000      4.00000
    =====================================
     EIGENVALUE:           EIGENVECTOR
      10.90699   :      .368     .647    .668
        .80990   :    -.903      .076    .424
      -2.71690   :      .223    -.759    .612
```

15. 求实矩阵全部特征值和特征向量的QR方法

15.1 功能

求n阶矩阵A的全部特征值或全部特征值和特征向量.

先用初等相似变换把实矩阵A化为上Hessenberg(赫申伯格)型矩阵，然后对Hessenberg阵采用带原点位移的两步QR 算法.

15.2 使用说明

1° 子程序调用语句

求矩阵A的全部特征值:

CALL QRVA (N,A, EPS, WR,WI,KNT)

求矩阵A的全部特征值和特征向量:

CALL QRVE (N, A, EPS, WR, WI, VE, KNT, D)

2° 虚元说明

输入参数：

N　　整变量，矩阵A的阶数

A　　N*N个元素的二维实数组，存放矩阵A的元素

EPS　实变量，存放该计算机所能表示的满足1+EPS>1的最小正数．它取决于计算机的有效位数

输出参数：

WR　　N个元素的一维实数组，存放矩阵A的特征值的实部

WI　　N个元素的一维实数组，存放矩阵A的特征值的虚部，其次序和WR数组中的实部相应

VE　　N＊N个元素的二维实数组，存放矩阵A的特征向量（未规格化），如果第i个特征向量是实的，则VE的第i列是其相应的特征向量．如果第i列和第$i+1$列是一对复共轭特征值，则VE的第i列和第$i+1$列分别是该对复共轭特征值中虚部为正的一个特征值所对应的特征向量的实部与虚部

KNT　N个元素的一维数组，表示计算每一个特征值所用的工作单元

D　　N个元素的一维数组

3° 附注

在计算过程中，若迭代30次仍得不到特征值，则停机，打印STOP 4444．

15.3 例题

例 1　求矩阵

$$A=\begin{pmatrix} 1 & 2 & 3 & 5 \\ 2 & 4 & 1 & 6 \\ 1 & 2 & -1 & 3 \\ 2 & 0 & 1 & 3 \end{pmatrix}$$

的全部特征值。

例 2 求矩阵

$$A=\begin{pmatrix} 3 & 1 & 2 & 5 \\ 2 & 1 & 3 & 7 \\ 3 & 1 & 2 & 4 \\ 4 & 1 & 3 & 2 \end{pmatrix}$$

的全部特征值和特征向量。

15.4 程序和计算结果

例 1

```
      SUBROUTINE QRVA (N, A, EPS, WR, WI, KNT)
      DIMENSION A (N, N), WR(N), WI(N),KNT(N)
      CALL EH (N, A, 1, N, KNT)
      CALL HQR (N, A, EPS, WR, WI, KNT)
      RETURN
      END

      SUBROUTINE EH (N, A, K, L, KNT)
      DIMENSION A (N, N), KNT(N)
      LA=L-1
      K1=K+1
      IF (K1. GT. LA) RETURN
      DO 80 M=K1, LA
      I=M
      X=0.0
      DO 10 J=M, L
      IF (ABS (A(J, M-1)). LE. ABS(X)) GO TO 10
      X=A(J, M-1)
      I=J
10    CONTINUE
      KNT(M)=I
      IF (I. EQ. M) GO TO 40
      J1=M-1
      DO 20 J=J1, N
      Y=A(I, J)
      A(I, J)=A(M, J)
20    A(M, J)=Y
      DO 30 J=1, L
```

```
      Y=A(J, I)
      A(J, I)=A(J, M)
30    A(J, M)=Y
40    IF(X. EQ. 0.0) GOTO 80
      I1=M+1
      DO 70 I=I1, L
      Y=A (I, M-1)
      IF(Y. EQ. 0.0) GOTO 70
      Y=Y/X
      A (I, M-1)=Y
      DO 50 J=M, N
50    A (I, J)=A(I, J)-Y*A(M, J)
      DO 60 J=1, L
60    A(J, M)=A(J, M)+Y*A(J, I)
70    CONTINUE
80    CONTINUE
      RETURN
      END


      SUBROUTINE HQR (NN, H, EPS, WR, WI, KNT)
      DIMENSION H (NN, NN), WR(NN), WI(NN), KNT(NN)
      N=NN
      T=0.
2     IF(N. EQ. 0) RETURN
      ITS=0
      NA=N-1
      IF (N. EQ. 1) GOTO 11
6     DO 10 L1=2, N
      L=N+2-L1
      IF (ABS(H(L, L-1)). LE. (EPS*(ABS(H(L-1, L-1))
     ++ABS (H(L, L))))) GOTO 12
10    CONTINUE
11    L=1
12    X=H (N, N)
      IF (L. EQ. N) GOTO 85
      Y=H (NA, NA)
      W=H (N, NA)*H (NA, N)
      IF (L. EQ. NA) GOTO 90
      IF (ITS. EQ. 30) STOP 4444
      IF (ITS. EQ. 10. OR. ITS. EQ. 20) GOTO 14
      GOTO 20
14    T=T+X
```

```
      DO 15 I=1, N
15    H (I, I)=H(I, I) -X
      S=ABS (H(N, NA))+ABS (H(NA, N-2))
      Y=0.75*S
      X=Y
      W=-0.4375*S*S
20    ITS=ITS+1
      N2=N-2
      DO 30 M1=L, N2
      M=N2+L-M1
      Z=H(M, M)
      R=X-Z
      S=Y-Z
      P= (R*S-W)/H(M+1, M)+H(M, M+1)
      Q=H(M+1, M+1)-Z-R-S
      R=H(M+2, M+1)
      S=ABS(P)+ABS(R)+ABS(Q)
      P=P/S
      Q=Q/S
      R=R/S
      IF (M. EQ. L) GOTO 35
      IF (ABS(H(M,M-1))*(ABS(Q)+ABS(R)).LE.(EPS*ABS(P)*
     +(ABS(H(M-1,M-1))+ABS(Z)+ABS(H(M+1,M+1))))) GOTO 35
30    CONTINUE
35    I1=M+2
      DO 36 I=I1, N
36    H (I, I-2)=0.
      I1=M+3
      IF (I1. GT. N) GOTO 39
      DO 38 I=I1, N
38    H(I, I-3)=0.
39    DO 80 K=M, NA
      IF(K. EQ. M) GOTO 40
      P=H(K, K-1)
      Q=H(K+1, K-1)
      R=0.
      IF (K. NE.NA) R=H(K+2, K-1)
      X=ABS(P)+ABS(Q)+ABS(R)
      IF (X. EQ. 0.) GOTO 80
      P=P/X
      Q=Q/X
      R=R/X
40    S=SQRT (P*P+Q*Q+R*R)
```

```
      IF (P. LT. 0.) S=-S
      IF (K. NE. M) H(K, K-1)=-S*X
      IF (K. EQ. M. AND.L. NE. M) H(K, K-1)=-H(K, K-1)
      P=P+S
      X=P/S
      Y=Q/S
      Z=R/S
      Q=Q/P
      R=R/P
      DO 50 J=K, N
      P=H(K, J) +Q*H (K+1, J)
      IF (K. EQ. NA) GOTO 45
      P=P+R*H (K+2, J)
      H(K+2, J)=H(K+2, J)-P*Z
45    H(K+1, J)=H(K+1, J)-P*Y
      H(K, J)=H(K, J)-P*X
50    CONTINUE
      J=N
      IF ((K+3). LT. N) J=K+3
      DO 60 I=L, J
      P=X*H(I, K)+Y*H(I, K+1)
      IF (K. EQ. NA) GOTO 55
      P=P+Z*H (I, K+2)
      H (I, K+2)=H(I, K+2)-P*R
55    H (I, K+1)=H(I, K+1)-P*Q
      H(I, K)=H(I, K)-P
60    CONTINUE
80    CONTINUE
      GOTO 5
85    WR(N)=X+T
      WI(N)=0.
      KNT(N)=ITS
      N=NA
      GOTO 2
90    P=(Y-X)/2.
      Q=P*P+W
      Y=SQRT(ABS(Q))
      KNT(N)=ITS
      KNT(NA)=ITS
      X=X+T
      IF(Q) 94, 94, 92
92    IF (P. LT. 0.) Y=-Y
      Y=P+Y
```

```
      WR(NA)=X+Y
      WR(N)=X-W/Y
      WI (NA)=0.
      WI(N)=0.
      GOTO 96
94    WR (NA) =X+P
      WR(N)=WR(NA)
      WI(NA)=Y
      WI(N)=-Y
96    N=N-2
      GOTO 2
      END
```

```
      DIMENSION A(4, 4), WR(4), WI(4), KNT(4)
      DATA A/1., 2., 1., 2., 2., 4., 2., 0., 3.,
     +1., -1., 1., 5., 6., 3., 3./
      CALL QRVA (4, A, 1E-6, WR, WI, KNT)
      WRITE (*, 11)
11    FORMAT (6X, 10X, 4HREAL, 9X, 8HIMAGINAL, 6X,
     +5HTIMES)
      WRITE (*, 12)  (WR(I), WI(I), KNT(I), I=1, 4)
12    FORMAT (6X, 2F15.5, I10)
      STOP
      END
               REAL          IMAGINAL        TIMES
             8.07159          .00000           0
             1.58793          .00000           0
            -1.32976          .32285           3
            -1.32976         -.32285           3
```

例 2

```
      SUBROUTINE QRVE (N, A, EPS, WR, WI, VE, KNT, D)
      DIMENSION A(N,N),WR(N),WI(N),VE(N,N), D(N), KNT(N)
      CALL BAL (N, A, LOW, IUP, D)
      CALL EH (N, A, LOW, IUP, KNT)
      CALL ES (N, A, LOW, IUP, KNT, VE)
      CALL HQRVE (N, A, EPS, LOW, IUP, WR, WI, VE, KNT)
      CALL BAB (N, LOW, IUP, D, VE)
      RETURN
      END
```

```
      SUBROUTINE BAL (N, A, LOW, IUP, D)
      DIMENSION A (N, N), D(N)
      B=2.
      B2=B * B
      L=1
      K=N
10    DO 15 J1=1, K
      J=K-J1+1
      R=0.
      DO 12 I=1, K
12    IF (I. NE. J) R=R+ABS (A(J, I))
      IF (R. NE. 0.) GOTO 15
      CALL EXC (K, N, J, K, L, D, A)
      IF (K. EQ. 1) GOTO 20
      K=K-1
      GOTO 10
15    CONTINUE
20    DO 25 J=L, K
      C=0.
      DO 22 I=L, K
22    IF (I. NE. J) C=C+ABS (A(I, J))
      IF (C. NE. 0.) GOTO 25
      CALL EXC (L, N, J, K, L, D, A)
      IF (L. EQ. K) GOTO 26
      L=L+1
      GOTO 20
25    CONTINUE
26    LOW=L
      IUP=K
      DO 27 I=L, K
27    D(I)=1.
28    NO=0
      DO 45 I=L, K
      C=0.
      R=0.
      DO 29 J=L, K
      IF (J. EQ. I) GOTO 29
      C=C+ABS (A(J, I))
      R=R+ABS (A(I, J))
29    CONTINUE
      G=R/B
      F=1.
      S=C+R
```

```
30    IF (C. GE. G) GOTO 35
      F=F * B
      C=C * B2
      GOTO 30
35    G=R * B
40    IF(C. LT. G) GOTO 41
      F=F/B
      C=C/B2
      GOTO 40
41    IF((C+R)/F. GE. 0.95 * S) GOTO 45
      G=1./F
      D(I)=D(I) * F
      NO=1
      DO 42 J=L, N
42    A(I, J)=A(I, J) * G
      DO 43 J=1, K
43    A(J, I)=A(J, I) * F
45    CONTINUE
      IF (NO. EQ.1) GOTO 28
      RETURN
      END

      SUBROUTINE EH (N, A, K, L, KNT)
      DIMENSION A (N, N), KNT(N)
      LA=L-1
      K1=K+1
      IF (K1. GT. LA) RETURN
      DO 80 M=K1, LA
      I=M
      X=0.
      DO 10 J=M, L
      IF (ABS(A(J, M-1)). LE. ABS(X)) GOTO 10
      X=A(J, M-1)
      I=J
10    CONTINUE
      KNT(M)=I
      IF (I. EQ. M) GOTO 40
      J1=M-1
      DO 20 J=J1, N
      Y=A(I, J)
      A(I, J)=A(M, J)
20    A(M, J)=Y
      DO 30 J=1, L
```

```
      Y=A(J, I)
      A(J, I)=A(J, M)
30    A(J, M)=Y
40    IF (X. EQ. 0.) GOTO 80
      I1=M+1
      DO 70 I=I1, L
      Y=A(I, M-1)
      IF (Y. EQ. 0.) GOTO 70
      Y=Y/X
      A (I, M-1)=Y
      DO 50 J=M, N
50    A (I, J)=A(I, J)-Y*A(M, J)
      DO 60 J=1, L
60    A(J, M)=A(J, M)+Y*A(J, I)
70    CONTINUE
80    CONTINUE
      RETURN
      END

      SUBROUTINE ES (N, A, LOW, IUP, KNT, VE)
      DIMENSION A(N, N), VE(N, N), KNT(N)
      IUP1=IUP-1
      LOW1=LOW+1
      DO 20 I=1, N
      DO 10 J=1, N
10    VE(I, J)=0.
20    VE(I, I)=1.
      IF(LOW1. GT. IUP1) RETURN
      DO 50 I1=LOW1, IUP1
      I=LOW1+IUP1-I1
      J=KNT(I)
      K1=I+1
      DO 30 K=K1, IUP
30    VE(K, I)=A(K, I-1)
      IF(I. EQ. J) GOTO 50
      DO 40 K=I, IUP
      VE(I, K)=VE(J, K)
40    VE(J, K)=0.
      VE(J, I)=1.
50    CONTINUE
      RETURN
      END
```

```
      SUBROUTINE EXC (M, N, J, K, L, D, A)
      DIMENSION D(N), A(N, N)
      D(M)=FLOAT(J)
      IF (J. EQ. M) RETURN
      DO 10 I=1, K
      F=A(I, J)
      A(I, J)=A(I, M)
   10 A(I, M)=F
      DO 30 I=L, N
      F=A(J, I)
      A(J, I)=A(M, I)
   30 A(M, I) =F
      RETURN
      END


      SUBROUTINE CDIV (XR, XI, YR, YI, ZR, ZI)
      IF (ABS(YR). LE. ABS(YI)) GOTO 10
      H=YI/YR
      G=H * YI+YR
      ZR=(XR+H * XI)/G
      ZI=(XI-H * XR)/G
      RETURN
   10 H=YR/YI
      G=H * YR+YI
      ZR=(H * XR+XI)/G
      ZI=(H * XI-XR)/G
      RETURN
      END


      SUBROUTINE BAB (N, LOW, IUP, D, VE)
      DIMENSION D(N), VE(N, N)
      DO 10 I=LOW, IUP
      S=D(I)
      DO 10 J=1, N
   10 VE(I, J)=VE(I, J) * S
      DO 40 I1=1, N
      IF (I1. GE. LOW. AND. I1. LE. IUP) GOTO 40
      IF (I1. LT. LOW) I=LOW-I1
      IF (I1. GT. IUP) I=I1
      IF (FLOAT(I). EQ. D(I)) GOTO 40
      DO 30 J=1, N
```

```
      S=VF (I, J)
      VE (I, J)=VE(K, J)
30    VE (K, J)=S
40    CONTINUE
      RETURN
      END

      SUBROUTINE HQRVE(N,A,EPS,LOW,IUP,WR,WI,VE,KNT)
      DIMENSION A (N, N), WR(N), WI(N), VE(N, N), KNT(N)
      DO 10 I=1, N
      IF (I. GE. LOW. AND. I. LE. IUP) GOTO 10
      WR(I)=A(I, I)
      WI(I)=0.
      KNT(I)=0.
10    CONTINUE
      NE=IUP
      T=0.
16    IF (NE. LT. LOW) GOTO 220
      ITS=0
      NA=NE-1
20    LOW1=LOW+1
      IF (NE. LT. LOW1) GOTO 26
      DO 25 L1=LOW1, NE
      L=NE+LOW1-L1
      IF (ABS (A(L, L-1)). LE. EPS*(ABS(A(L-1,
     +L-1))+ABS(A(L, L)))) GOTO 30
25    CONTINUE
26    L=LOW
30    X=A(NE, NE)
      IF (L. EQ. NE) GOTO 150
      Y=A(NA, NA)
      W=A(NA, NE)*A(NE, NA)
      IF (L. EQ. NA) GOTO 160
      IF (ITS. EQ. 30) STOP 4444
      IF (ITS. EQ. 10. OR. ITS. EQ. 20) GOTO 40
      GOTO 50
40    T=T+X
      DO 45 I=LOW, NE
45    A(I, I)=A(I, I)-X
      S=ABS(A(NE, NA))+ABS(A(NA, NE-2))
      X=0.75*S
      Y=X
```

```
      W=-0.4375*S*S
50    ITS=ITS+1
      NE1=NE-2
      DO 60 M1=L, NE1
      M=NE1+L-M1
      Z=A(M, M)
      R=X-Z
      S=Y-Z
      P=(R*S-W)/A(M+1, M)+A(M, M+1)
      Q=A(M+1, M+1)-Z-R-S
      R=A(M+2, M+1)
      S=ABS(P)+ABS(Q)+ABS(R)
      P=P/S
      Q=Q/S
      R=R/S
      IF(M. EQ. L) GOTO 65
      IF (ABS(A(M, M-1))*(ABS(Q)+ABS(R)). LE. EPS*
     +ABS(P)*(ABS(A(M-1, M-1))
     ++ABS(Z)+ABS(A(M+1, M+1)))) GOTO 65
60    CONTINUE
65    M2=M+2
      DO 70 I=M2, NE
70    A(I, I-2)=0.
      M2=M+3
      IF(M2. GT. NE) GOTO 76
      DO 75 I=M2, NE
75    A(I, I-3)=0.
76    DO 140 K=M, NA
      IF(K. EQ. M) GOTO 80
      P=A(K, K-1)
      Q=A(K+1, K-1)
      R=0.
      IF(K. NE. NA) R=A(K+2, K-1)
      X=ABS(P)+ABS(Q)+ABS(R)
      IF(X. EQ. 0.) GOTO 140
      P=P/X
      Q=Q/X
      R=R/X
80    S=SQRT(P*P+Q*Q+R*R)
      IF (P. LT. 0.) S=-S
      IF(K. NE. M.) A(K, K-1)=-S*X
      IF(K. EQ. M. AND. L. NE. M) A(K, K-1)=-A(K, K-1)
      P=P+S
```

```
      X=P/S
      Y=Q/S
      Z=R/S
      Q=Q/P
      R=R/P
      DO 100 J=K, N
      P=A(K, J)+Q*A(K+1, J)
      IF (K. EQ. NA) GOTO 90
      P=P+R*A(K+2, J)
      A(K+2, J)=A(K+2, J)-P*Z
 90   A(K+1, J)=A(K+1, J)-P*Y
      A(K, J)=A(K, J)-P*X
100   CONTINUE
      J=NE
      IF((K+3). LT. NE) J=K+3
      DO 120 I=1, J
      P=X*A(I, K)+Y*A(I, K+1)
      IF(K. EQ. NA) GOTO 110
      P=P+Z*A(I, K+2)
      A(I, K+2)=A(I, K+2)-P*R
110   A(I, K+1)=A(I, K+1)-P*Q
      A(I, K)=A(I, K)-P
120   CONTINUE
      DO 130 I=LOW, IUP
      P=X*VE(I, K)+Y*VE(I, K+1)
      IF (K. EQ. NA) GOTO 125
      P=P+Z*VE(I, K+2)
      VE(I, K+2)=VE(I, K+2)-P*R
125   VE(I, K+1)=VE(I, K+1)-P*Q
      VE(I, K)=VE(I, K)-P
130   CONTINUE
140   CONTINUE
      GOTO 20
150   WR(NE)=X+T
      A(NE, NE)=WR(NE)
      WI(NE)=0.
      KNT(NE)=ITS
      NE=NA
      GOTO 15
160   P=(Y-X)/2.
      Q=P*P+W
      Z=SQRT(ABS(Q))
      A(NE, NE)=X+T
```

```
      X=A(NE, NE)
      A(NA, NA)=Y+T
      KNT(NE)=ITS
      KNT(NA)=ITS
      IF(Q. LE. 0.) GOTO 200
      Z=P+SIGN(Z, P)
      WR(NA)=X+Z
      WR(NE)=WR(NA)
      IF (Z. NE. 0.) WR(NE)=X-W/Z
      WI(NA)=0.
      WI(NE)=0.
      X=A(NE, NA)
      R=SQRT (X*X+Z*Z)
      P=X/R
      Q=Z/R
      DO 170 J=NA, N
      Z=A(NA, J)
      A(NA, J)=Z*Q+P*A(NE, J)
      A(NE, J)=Q*A(NE, J)-P*Z
170   CONTINUE
      DO 180 I=1, NE
      Z=A(I, NA)
      A(I, NA)=Q*Z+P*A(I, NE)
      A(I, NE)=Q*A(I, NE)-P*Z
180   CONTINUE
      DO 190 I=LOW, IUP
      Z=VE(I, NA)
      VE(I, NA)=Q*Z+P*VE(I, NE)
190   VE(I, NE)=Q*VE(I, NE)-P*Z
      GOTO 210
200   WR(NA)=X+P
      WR(NE)=WR(NA)
      WI(NA)=Z
      WI(NE)=-Z
210   NE=NE-2
      GOTO 15
220   SN=0.
      K=1
      DO 240 I=1, N
      DO 230 J=K, N
230   SN=SN+ABS(A(I, J))
240   K=I
      DO 400 NE1=1, N
```

```
      NE=N+1-NE1
      P=WR(NE)
      Q=WI(NE)
      NA=NE-1
      IF(Q) 290, 245, 400
245   M=NE
      A(NE, NE)=1.
      IF(NA. EQ.0) GOTO 400
      DO 280 I1=1, NA
      I=NA+1-I1
      W=A(I, I)-P
      R=A(I, NE)
      IF (M. GT.NA) GOTO 255
      DO 250 J=M, NA
250   R=R+A(I, J)*A(J, NE)
255   IF (WI(I). GE. 0.) GOTO 260
      Z=W
      S=R
      GOTO  280
260   M=I
      IF (WI(I). NE. 0.) GOTO 270
      WW=W
      IF (W. EQ. 0.) WW=EPS*SN
      A(I, NE)=-R/WW
      GOTO 280
270   X=A(I, I+1)
      Y=A(I+1, I)
      Q=(WR(I)-P)**2+WI(I)**2
      T=(X*S-Z*R)/Q
      A(I, NE)=T
      IF(ABS(X). GT. ABS(Z)) A(I+1, NE)=(-R-W*T)/X
      IF (ABS(X). LE. ABS(Z)) A(I+1, NE)=(-S-Y*T)/Z
280   CONTINUE
      GOTO 400
290   M=NA
      IF(ABS(A(NE, NA)). LE. ABS(A(NA, NE)))  GOTO 300
      A(NA, NA)=-(A(NE, NE)-P)/A(NE, NA)
      A(NA, NE)=-Q/A(NE, NA)
      GOTO 305
300   CALL CDIV(-A(NA, NE), 0., A(NA, NA)-P, Q,A(NA,NA)
     +, A(NA, NE))
305   A(NE, NA)=1.
      A(NE, NE)=0.
```

```
      NA1=NA-1
      IF(NA1. EQ. 0) GOTO 400
      DO 390 I1=1, NA1
      I=NA-I1
      W=A(I, I)-P
      RA=A(I, NE)
      SA=0.
      DO 310 J=M, NA
      RA=RA+A(I, J)*A(J, NA)
310   SA=SA+A(I, J)*A(J, NE)
      IF(WI(I)) 320, 330, 340
320   Z=W
      R=RA
      S=SA
      GOTO 390
330   M=I
      CALL CDIV(-RA, -SA, W, Q, A(I, NA), A(I, NE))
      GOGT 390
340   M=I
      X=A(I, I+1)
      Y=A(I+1, I)
      VR=(WR(I)-P)**2+WI(I)**2-Q**2
      VI=(WR(I)-P)*2.*Q
      IF(VR. EQ. 0.. AND. VI. EQ. 0.) VR=EPS*SN*(ABS(W)
     ++ABS(Q)+ABS(X)+ABS(Y)+ABS(Z))
      CALL CDIV(X*R-Z*RA+Q*SA,X*S-Z*SA-Q*RA,VR,VI
     +, A(I, NA), A(I, NE))
      IF(ABS(X). LE. (ABS(Z)+ABS(Q))) GOTO 350
      A(I+1, NA)=(-RA-W*A(I, NA)+Q*A(I, NE))/X
      A(I+1, NE)=(-SA-W*A(I, NE)-Q*A(I, NA))/X
      GOTO 390
350   CALL CDIV(-R-Y*A(I, NA), -S-Y*A(I, NE), Z,
     +Q, A(I+1, NA), A(I+1, NE))
390   CONTINUE
400   CONTINUE
      DO 420 I=1, N
      IF(I. GE. LOW. AND. I. LE. IUP) GOTO 420
      I1=I+1
      DO 410 J=I1, N
410   VE(I, J)=A(I, J)
420   CONTINUE
      DO 500 J1=LOW, N
      J=N+LOW-J1
```

```
      M=J
      IF(J. GT. IUP) M=IUP
      L=J-1
      IF(WI(J)) 430, 460, 500
430   DO 450 I=LOW, IUP
      Y=0.
      Z=0.
      DO 440 K=LOW, M
      Y=Y+VE(I, K)*A(K, L)
440   Z=Z+VE(I, K)*A(K, J)
      VE(I, L)=Y
450   VE(I, J)=Z
      GOTO 500
460   DO 480 I=LOW, IUP
      Z=0.
      DO 470 K=LOW, M
470   Z=Z+VE(I, K)*A(K, J)
480   VE(I, J)=Z
500   CONTINUE
      RETURN
      END
```

```
      DIMENSION A(4, 4), WR(4), WI(4), VE(4, 4), KNT(4), D(4)
      DATA A/3., 2., 3., 4., 1., 1., 1., 1., 2., 3.,
     +2., 3., 5., 7., 4., 2./
      CALL QRVE (4, A, 1E-6, WR, WI, VE, KNT, D)
      WRITE(*, 11)
11    FORMAT(1X, 9X, 4HREAL, 10X, 8HIMA
     +GINAL, 6X, 5HTIMES)
      WRITE(*, 12) (WR(I), WI(I), KNT(I), I=1, 4)
12    FORMAT(1X, 2F15.5, I10)
      WRITE (*, 15)
15    FORMAT (8X, 36 (1H=))
      WRITE (*, 13)
13    FORMAT (3X, 28HTHE CHARACTERIST VORTORS ARE)
      WRITE (*, 14)  ((VE(I, J), J=1, 4), I=1, 4)
14    FORMAT (6X, 4F8.4)
      STOP
      END
```

REAL	IMAGINAL	TIMES
10.59198	.00000	0
−2.36630	.00000	0
.19135	.00000	3
−.41703	.00000	3

==

THE CHARACTERIST VORTORS ARE

−.7123	.3661	.5316	.0289
−.8302	1.0164	−1.8018	−2.5744
−.6504	.0976	−.1917	.0208
−.6553	−.6352	.1384	.2408

16. 解非线性方程组的Brown方法

16.1 功能

本程序用 Brown 方法求 n 个变量的 n 个非线性方程组

$$F(x)=0$$

的一组实根 $x^*=(x_1^*, x_2^*, \cdots, x_n^*)^T$，其中 $F: R^n \to R^n$。

16.2 使用说明

1° 子程序调用语句

CALL NONLIN (N, NUMSIG, MAXIT, IPRINT, X, EPS)

2° 虚元说明

输入参数：

N　　整变量，被求方程组的阶数，也是未知量的个数

NUMSIG　整变量，有效位数

EPS　　实变量，表示迭代精度，当 $\|F(x)\|_\infty \leq$ EPS 时迭代终止

输入兼输出参数：

X　　N 个元素的一维实数组，输入时表示初始近似向量，输出时为方程组的解向量。

MAXIT　整变量，输入时表示允许的迭代次数，输出时表示使

用的迭代次数

IPRINT　整变量，=1表示选择输出。但是当迭代次数大于 MAXIT 或 Jacobi 矩阵奇异时，它总是输出

3°　附注

本程序适用于 N≤30 的方程组，占用工作单元为 (2N+5)*N。当 N>30 时，应修改程序中有关工作单元长度才能使用。另外，还应编制子程序 SUBROUTINE AUXFCN(X, Y, K)。

16.3 例题

求方程组

$$
\begin{cases}
f_1(\boldsymbol{x})=\dfrac{1}{3}\left(x_1x_4+\dfrac{1}{2}x_2x_4-\dfrac{1}{4}x_4^3\right)=0\\[2ex]
f_2(\boldsymbol{x})=\dfrac{1}{3}\Big[x_2+\dfrac{1}{2}x_1x_4+\dfrac{1}{2}(1-x_4)x_3\\
\qquad\qquad -\dfrac{1}{4}(x_4^2+x_4+1)\Big]=0\\[2ex]
f_3(\boldsymbol{x})=\dfrac{1}{3}\Big[x_3(1-x_4)+\dfrac{1}{2}x_2(1-x_4)\\
\qquad\qquad +\dfrac{1}{4}(x_4^3+x_4^2+x_4-3)\Big]=0\\[2ex]
f_4(\boldsymbol{x})=(x_3-x_2)(2x_4+1)+(x_1-x_3)(2x_1+2x_2+2x_3-3x_4^2)\\
\qquad =0
\end{cases}
$$

的解，取初始近似 $\boldsymbol{x}^0=(0,\ 0.01,\ 1,\ 0.75)^T$。

16.4 程序与计算结果

```
      SUBROUTINE NONLIN(N,NUMSIG,MAXIT,IPRINT,X, EPS)
      REAL X(30), PART(30), TEMP(30), COE(30, 31),
     +RELCON, F, FACTOR, HOLD, H, FPLUS, DERMAX, TEST
      DIMENSION ISUB(30), LOOKUP(30, 30)
C     DELTA WILL BE A FUNCTION OF THE MACHINE AND
C     THE PRECISION USED:
```

```
      DELTA=1.E-7
      RELCON=10.E+0**(-NUMSIG)
      JTEST=1
      IF(IPRINT.EQ.1) WRITE (*, 48)
    4 FORMAT(1X/)
      WRITE(*, 47) (I, I=1, N)
   47 FORMAT(10X,' M1', 8X, 4('X(', I1,')', 8X))
      DO 700 M=1, MAXIT
      IQUIT=0
      FMAX=0.
      M1=M-1
      IF (IPRINT.NE.1) GO TO 9
      WRITE (*, 49) M1, (X(I), I=1, N)
   49 FORMAT (10X, I1, 2X, 4E13.6/)
    9 DO 10 J=1, N
   10 LOOKUP (1, J)=J
C     THE ARRAY LOOKUP PERMITS A PARTIAL PIVOTING
C     EFFECT WITHOUT HAVING TO PHYSICALLY
C     INTERCHANGE ROWS OR COLUMNS
      DO 500 K=1, N
      IF (K-1) 134, 134, 131
  131 KMIN=K-1
      CALL BACK (KMIN, N, X, ISUB, COE, LOOKUP)
C     SET UP PARTIAL DERIVATIVES OF KTH FUNCTION.
  134 CALL AUXFCN (X, F, K)
      FMAX=AMAX1 (FMAX, ABS(F))
      IF (ABS(F).GE.EPS) GOTO 1345
      IQUIT=IQUIT+1
      IF (IQUIT.NE.N) GOTO 1345
      GO TO 725
 1345 FACTOR=0.001E+00
  135 ITALLY=0
      DO 200 I=K, N
      ITEMP=LOOKUP (K, I)
      HOLD=X (ITEMP)
      PREC=5.E-6
C     PREC IS A FUNCTION OF THE MACHING SIGNIFICANCE,
C     SIG, AND SHOULD BE COMPUTED AS
C     PREC+5.*10.**(-SIG+2). IN THIS INSTANCE
C     WE WERE DEALING WITH AN 8 DIGIT MACHING.
      ETA=FACTOR*ABS (HOLD)
      H=AMIN1 (FMAX, ETA)
      IF (H.LT.PREC) H=PREC
```

```
      X (ITEMP) =HOLD+H
      IF (K-1) 161, 161, 151
  151 CALL BACK (KMIN, N, X, ISUB, COE, LOOKUP)
  161 CALL AUXFCN (X, FPLUS, K)
      PART (ITEMP)=(FPLUS-F)/H
      X (ITEMP)=HOLD
      IF (ABS(PART(ITEMP)). LT. DELTA) GOTO 190
      IF (ABS(F/PART(ITEMP)). LE. 1.E+15) GOTO 200
  190 ITALLY=ITALLY+1
  200 CONTINUE
      IF (ITALLY. LE. (N-K)) GOTO 202
      FACTOR=FACTOR * 10. 0E+00
      IF (FACTOR. GT. 11.) GOTO 775
      GO TO 135
  202 IF (K. LT. N) GOTO 203
      IF (ABS(PART(ITEMP)). LT. DELTA) GOTO 775
      COE (K, N+1)=0.0
      KMAX=ITEMP
      GO TO 500
C     FIND PARTIAL DERIVATIVE OF LARGEST ABSOLUTE
C     VALUE
  203 KMAX=LOOKUP (K, K)
      DERMAX=ABS (PART (KMAX))
      KPLUS=K+1
      DO 210 I=KPLUS, N
      JSUB=LOOKUP (K, I)
      TEST=ABS (PART(JSUB))
      IF (TEST. LT. DERMAX) GO TO 209
      DERMAX=TEST
      LOOKUP (KPLUS, I)=KMAX
      KMAX=JSUB
      GO TO 210
  209 LOOKUP (KPLUS, I)=JSUB
  210 CONTINUE
      IF (ABS(PART(KMAX)). EQ. 0.0) GO TO 775
C     SET UP COEFFICIENNTS FOR KTH ROW OF TRIANGULAR
C     LINEAR SYSTEM USED TO BACK-SOLVE FOR THE FIRST
C     K VALUES OF X(I)
      ISUB(K)=KMAX
      COE(K, N+1)=0. 0E+00
      DO 220 J=KPLUS, N
      JSUB=LOOKUP (KPLUS, J)
      COE(K, JSUB)=-PART (JSUB)/PART (KMAX)
```

```
      COE(K, N+1)=COE (K, N+1)+PART (JSUB)*X (JSUB)
  220 CONTINUE
  500 COE(K, N+1)=(COE(K, N+1)-F)/PART(KMAX)+X(KMAX)
C     BACK SUBSTITUTE TO OBTAIN NEXT APPROXIMATION
C     TO X:
      X(KMAX)=COE(N, N+1)
      IF (N. EQ. 1) GO TO 610
      CALL BACK (N-1, N, X, ISUB, COE, LOOKUP)
  610 IF (M-1) 650, 650, 625
C     TEST FOR CONVERGENCE
  625 DO 630 I=1, N
      IF (ABS(TEMP(I)-X(I)). GT. ABS(X(I))*RELCON)
     +GO TO 649
  630 CONTINUE
      JTEST=JTEST+1
      IF (JTEST-3) 650, 725, 725
  649 JTEST=1
  650 DO 660 I=1, N
  660 TEMP(I)=X(I)
  700 CONTINUE
      WRITE (*, 1753)
 1753 FORMAT (5X,/' NO CONVERGENCE. MAXIMUM NUMBER
     +OF ITERATIONS USED.')
      IF (IPRINT. NE. 1) GO TO 800
      WRITE(*, 1763)
 1763 FORMAT (5X,' FUNCTION VALUES AT THE LAST
     +APPROXIMATION FOLLOW:'/)
      IFLAG=1
      GO TO 7777
  725 IF (IPRINT. NE. 1) GO TO 800
 7777 DO 750 K=1, N
      CALL AUXFCN (X, PART(K), K)
  750 CONTINUE
      IF (IFLAG. NE. 1) GO TO 8777
      WRITE (*, 7788)  (PART(K), K=1, N)
 7788 FORMAT (10X, 4E12.6)
      GO TO 800
 8777 WRITE (*, 751)
  751 FORMAT (1X,//, 10X,' CONVERGENCE HAS BEEN
     +ACHIEVED. THE FUNCTION VALUES')
      WRITE (*, 7515)  (PART(K), K=1, N)
 7515 FORMAT(10X,' AT THE FINAL APPROXIMATION  FOLLOW:'
     +//, 10X, (4E13.6))
```

```
      GO TO 800
  775 WRITE (*, 752)
  752 FORMAT (5X, //' MODIFIED JACOBIAN IS SINGULAR.
     +TR Y A DIFFERENT')
      WRITE (*, 7525)
 7525 FORMAT (5X,' INITIAL APPROXIMATION.')
  800 MAXIT=M1+1
      RETURN
      END


      SUBROUTINE BACK (KMIN, N, X, ISUB, COE, LOOKUP)
C     THIS SUBROUTINE BACK—SOLVES THE FIRST KMIN  ROWS
C     OF A TRIANGUIARIZED LINEAR SYSTEMFOR IMPROVED X
C     VALUES IN TERMS PREVIOUS ONES
      DIMENSION  X(30), COE (30, 31), ISUB(30), LOOKUP(30, 30)
      DO 200 KK=1, KMIN
      KM=KMIN-KK+2
      KMAX=ISUB (KM-1)
      X (KMAX)=0.0
      DO 100 J=KM, N
      JSUB=LOOKUP (KM, J)
      X (KMAX)=X (KMAX) +COE (KM-1, JSUB) * X(JSUB)
  100 CONTINUE
      X (KMAX)=X(KMAX)+COE (KM-1, N+1)
  200 CONTINUE
      RETURN
      END


C     SAMPLE AUXFCN FOLLOWS
      SUBROUTINE AUXFCN(X, Y, K)
      DIMENSION X(4)
      T=X(4)
      GO TO (1, 2, 3, 4), K
    1 Y=(T/3.) * X(1)+(T/6.) * X(2)-T * * 3/12.
      RETURN
    2 Y=(T/6.) * X(1)+X(2)/3.+(1.-T) * X(3)/6.
     +-(T * * 2+T+1.)/12.
      RETURN
    3 Y=(1.-T) * X(2)/6.+(1.-T) * X(3)/3.+(T * * 3
     ++T * * 2+T-3.)/12.
```

```
      RETURN
4     Y=3.*(X(3)-X(1))*T**2+2.*(X(3)-X(2))*T+X(3)
     +-X(2)+2.*(X(1)**2-X(3)**2)+2.*X(2)*(X(1)-X(3))
      RETURN
      END

C     SAMPLE CALLING PROGRAM
      DIMENSION X(4)
      X(1)=0.0
      X(2)=0.01
      X(3)=1.
      X(4)=0.75
      CALL NONLIN (4, 6, 25, 1, X, 1.E-7)
      STOP
      END
C     END SAMPLE CALLING PROGRAM
```

MI	X(1)	X(2)	X(3)	X(4)
0	.00000E+00	.10000E-01	.10000E+01	.75000E+00
1	-.12950E+00	.13289E-01	.89895E+00	.27613E+00
2	.57903E-01	.10605E+00	.94352E+00	.41630E+00
3	-.12433E+00	.35808E+00	.99410E+00	.65690E+00
4	-.59370E-01	.21441E+00	.95860E+00	.51041E+00
5	-.42211E-01	.20898E+00	.95844E+00	.50056E+00
6	-.41666E-01	.20833E+00	.95833E+00	.49999E+00
7	-.41666E-01	.20833E+00	.95833E+00	.50000E+00

CONVERGENCE HAS BEEN ACHIEVED. THE FUNCTION VALUES AT THE FINAL APPROXIMATION FOLLOW:

.18626E-08   .00000E+00   .00000E+00   -.89407E-07

17. 解非线性方程组的割线法

17.1 功能

本程序用改进的 n 点割线法求 n 元非线性方程组

$$F(x)=0$$

的一组实根 $x^*=(x_1^*, x_2^*, \cdots, x_n^*)^T$，其中 F：$R^n \to R^n$。

17.2 使用说明

1° 子程序调用语句

CALL MSD(X, A, B, Y, Z, F, E, C, N, S, K, Q)

2° 虚元说明

输入参数：

N 整变量，方程组的阶数，也是未知数个数

S 实变量，误差控制参数，当 $\|F(\cdot)\|_\infty \leqslant S$ 时，迭代终止

输入兼输出参数：

X N个元素的一维实数组，输入时存放初始近似向量，输出时存放计算结果

输出参数：

Q 实变量，存放 $\|F(x)\|_\infty$

K 整变量，条件码，正常结束时，K为迭代步数。非正常结束时有以下标志：

K=-1000 表示输入错误

K=-2000 表示 $\|F(x)\|_\infty > 10^8$

K=-3000 表示迭代次数超过 40 * N

工作单元：

A, B	N * N 个元素的二维实数组
Y, Z, F, E, C	N个元素的一维实数组

3° 附注

使用本程序时，还需编制子程序 FNC

17.3 例题

求方程组

$$\begin{cases} f_1(x)=\dfrac{1}{2}\sin(x_1x_2)-\dfrac{x_2}{4\pi}-\dfrac{1}{2}x_1 \\ f_2(x)=\left(1-\dfrac{1}{4\pi}\right)(e^{2x_1}-e)+e\left(\dfrac{x_2}{\pi}-2x_1\right) \end{cases}$$

在 $x^0=(0.5,\ 3)^T$ 附近的解

17.4 程序和计算结果

```
      SUBROUTINE MSD (X, A, B, Y, Z, F, E, C, N, S, K, Q)
      DIMENSION X(N), A(N, N), Y(N), Z(N), F(N), E(N),
     +B(N, N), C(N)
      IF(N. LE. 0) GOTO 111
      K0=40*N
      CALL FNC (E, X, N)
      CALL NORMA (E, Q, N)
      EP=1.E-10
      IF (Q. LE. S) GOTO 222
      H=0.1
      DO 33 J=1, N
      DO 3 I=1, N
      IF(I-J) 1, 2, 1
1     Y(I)=X(I)
      GOTO 3
2     Y(I)=X(I)+H
3     CONTINUE
      CALL FNC (F, Y, N)
      DO 33 I=1, N
      A(I, J)=(F(I)-E(I))/H
33    B(I, J)=A(I, J)
      CALL IVSNC (B, C, Z, Y, F, N, EP)
      Q1=Q
      K=0
      GOTO 75
10    Q1=Q
      K=K+1
15    H=Q1
20    IF (H. LT. 0.1) GOTO 30
      H=H/2.
      GOTO 20
30    IF (K. EQ. K0) GOTO 555
      IF (K-N) 34, 32, 35
34    J0=K
      GOTO 40
32    J0=N
      GOTO 40
35    J0=MOD (K, N)
      IF (J0. EQ. 0) J0=N
40    DO 43 I=1, N
      IF (I. EQ. J0) GOTO 44
      Y(I)=X(I)
```

```
      GOTO 43
44    Y(I)=X(I)+H
43    CONTINUE
      CALL FNC (F, Y, N)
      CALL NORMA (F, Q0, N)
      IF (Q0. LT. S) GOTO 444
      DO 55 I=1, N
55    A(I, J0)=1./H*(F(I)-E(I))
      DO 50 J=1, N
      DO 50 I=1, N
50    B(I, J)=A(I, J)
      CALL IVSNC (B, C, Z, Y, F, N, EP)
75    CALL MIV (B, E, Z, N)
      G=1.
80    DO 85 I=1, N
85    Y(I)=X(I)-G*Z(I)
      CALL FNC(E, Y, N)
      CALL NORMA(E, Q, N)
      DO 77 I=1, N
77    C(I)=G*Z(I)
      CALL NORMA (C, Q0, N)
      IF (Q. LT. S) GOTO 101
      IF (Q. LT. Q1) GOTO 101
      IF (Q0. LT. (1. E4*S)) GOTO 101
      G=G/2.
      GOTO 80
101   DO 100 I=1, N
100   X(I)=Y(I)
      IF (Q. LT. S) GOTO 222
      IF (Q. GT. 1. E8) GOTO 333
      GOTO 10
444   Q=Q0
      DO 454 I=1, N
454   X(I)=Y(I)
      GOTO 222
555   K=-3000
      GOTO 222
111   K=-1000
      GOTO 222
333   K=-2000
222   CONTINUE
      RETURN
      END
```

```
      SUBROUTINE NORMA (X, W, N)
      DIMENSION X(N)
      W=0.
      DO 1 I=1, N
      IF (ABS (X(I)). LE. ABS (W)) GOTO 1
      W=ABS (X(I))
    1 CONTINUE
      RETURN
      END
```

```
      SUBROUTINE IVSNC (A, B, C, E, F, N, EP)
      DIMENSION A(N, N), B(N), C(N), E(N), F(N)
      DO 10 K=1, N
      Y=0.
      DO 20 I=K, N
      DO 20 J=K, N
      IF (ABS(A(I, J)). LE. ABS(Y)) GOTO 20
      Y=A(I, J)
      I2=I
      J2=J
   20 CONTINUE
      IF (ABS(Y). LT. EP) GOTO 32
      IF (I2. EQ. K) GOTO 33
      DO 11 J=1, N
      W=A(I2, J)
      A(I2, J)=A(K, J)
   11 A(K, J2)=W
   33 IF(J2. EQ. K) GOTO 44
      DO 22 I=1, N
      W=A(I, J2)
      A(I, J2)=A(I, K)
   22 A(I, K)=W
   44 E(K)=FLOAT(I2)
      F(K)=FLOAT(J2)
      DO 50 J=1, N
      IF(J-K) 2, 3, 2
    3 B(J)=1./Y
      C(J)=1.
      GOTO 4
    2 B(J)=-A(J, K)/Y
      C(J)=A(J, K)
    4 A(K, J)=0.
```

```
      A(J, K)=0.
50    CONTINUE
      DO 40 I=1, N
      DO 40 J=1, N
40    A(I, J)=A(I, J)+C(I)*B(J)
10    CONTINUE
      DO 60 L=1, N
      K=N-L+1
      K1=IFIX(E(K))
      K2=IFIX(F(K))
      IF(K1. EQ. K) GOTO 70
      DO 55 I=1, N
      W=A(I, K1)
      A(I, K1)=A(I, K)
55    A(I, K)=W
70    IF(K2. EQ. K) GOTO 60
      DO 66 J=1, N
      W=A(K2, J)
      A(K2, J)=A(K, J)
66    A(K, J)=W
60    CONTINUE
      RETURN
32    EP=-EP
      RETURN
      END


      SUBROUTINE MTV (A, B, C, N)
      DIMENSION A (N, N), B(N), C(N)
      DO 1 I=1, N
      C(I)=0.
      DO 1 J=1, N
1     C(I)=C(I)+A(I, J)*B(J)
      RETURN
      END


      DIMENSION A(2, 2), B(2, 2), X(2), Y(2), Z(2), E(2),
     +F(2), C(2)
      DATA X/0.5, 3./
      S=1.E-6
      N=2
      CALL MSD (X, A, B, Y, Z, F, E, C, N, S, K, Q)
```

```
      IF (EP. LT. 0.) WRITE (*, 3) EP
      WRITE (*, 2)K
      WRITE (*, 6)
      WRITE(*, 4) X
    2 FORMAT (15X,'K=', I5)
    3 FORMAT (1X,/, 15X,' EP=', E15.5)
    6 FORMAT (1X,/, 18X,' X(1)', 12X,' X(2)')
    4 FORMAT (9X, 2F16.8)
    5 FORMAT (1X, /, 15X,' ERROR=', E15.5)
      STOP
      END


      SUBROUTINE FNC (F, X, N)
      DIMENSION F(N), X(N)
      F(1)=1./2.*SIN(X(1)*X(2))-1./4./3.1415926*X(2)
     +-0.5*X(1)
      F(2)=(1.-1./4/3.1415926)*(EXP(2.*X(1))-EXP(1.))
     ++EXP(1.)*(1./3.1415926*X(2)-2.*X(1))
      RETURN
      END
```

```
K=8
          X(1)               X(2)
        .49999950         3.14159200
        ERROR=            .28312E-06
```

18. 解非线性方程组的Broyden方法

18.1 功能

本程序用逆 Broyden（布罗依登）秩 1 公式(9.7.13)求非线性方程组

$f_i(x_1, x_2, \cdots, x_n)=0 \qquad (i=1,2,\cdots,n)$

的一组实根 $x^*=(x_1^*, x_2^*, \cdots, x_n^*)$.

18.2 使用说明

1° 子程序调用语句

CALL SNSE4(N, EPS, MAXF,X, F, IT,IC, P, U,Y, H)

2° 虚元说明

输入参数：

N　　　整变量，方程的个数

MAXF　整变量，允许的最大迭代次数

EPS　　实变量，控制精度，$\|F(x)\|_\infty \leq$ EPS时迭代终止

输入兼输出参数：

X　　　N个元素的一维实数组，输入时为初始近似，输出时为计算结果

输出参数：

F　　　N个元素的一维实数组，存放f_i ($i=1, 2, \cdots, n$)的值

IT　　　整变量，条件码，迭代正常结束时IT为正整数．非正常结束时，IT为以下值：

IT＝－1　表示迭代次数≥MAXF

IT＝－2　表示$H_K \approx 0$ (K≥1)（见9.7.13）

IT＝－3　表示初始的$H \approx 0$

工作单元：

IC　　　　　整变量

P，U，Y　　N个元素的一维数组

H　　　　　N＊N个元素的二维实数组

18.3 例题

求非线性方程组

$$\begin{cases} f_1(x)=x_1-5x_2^2+7x_3^2+12=0 \\ f_2(x)=3x_1x_2+x_1x_3-11x_1=0 \\ f_3(x)=2x_2x_3+40x_1=0 \end{cases}$$

的一组实根．初始近似取为$x^0=(1.5, 7.5, -6.0)^T$．

18.4 程序和计算结果

```
SUBROUTINE SNSE4 (N, EPS, MAXF, X, F, IT, IC, P,
```

```
     +U, Y, H)
      DIMENSION X(N), F(N), P(N), U(N), Y(N), H(N, N)
      CALL FCT (X, F)
      IC=1
      DO 6 I=1, N
 6    P(I)=0.0
      DO 7 I=1, N
      DO 7 J=1, N
 7    H(I, J)=0.0
      DO 1 I=1, N
 1    H(I, I)=1.
      K=1
 2    IF(K. GT. N) GO TO 10
      P(K)=0.001
      L=-3
      GO TO 30
 31   P(K)=0.
      K=K+1
      GO TO 2
 10   DO 3 I=1, N
      SA=0.
      DO 4 J=1, N
 4    SA=SA-H(I, J)*F(J)
 3    P(I)=SA
      L=-2
      GO TO 30
 32   SA=0.
      DO 5 I=1, N
 5    SA=SA+F(I)*F(I)
      IF (SA. LT. EPS) GO TO 13
      IF (IC. GE. MAXF) GO TO 11
      GO TO 10
 13   IT=IC
 12   RETURN
 11   IT=-1
      GO TO 12
 30   DO 21 I=1, N
      X(I)=X(I)+P(I)
 21   U(I)=F(I)
      CALL FCT (X, F)
      IC=IC+1
      DO 22 I=1, N
 22   Y(I)=F(I)-U(I)
```

```
      SA=0.0
      DO 23 I=1, N
      SB=0.0
      DO 24 J=1, N
24    SB=SB+H(I, J)*Y(J)
      U(I)=SB-P(I)
23    SA=SA+SB*P(I)
      IF (ABS(SA). LT. 1.E-17) GO TO 28
      DO 25 J=1, N
      SB=0.
      DO 26 I=1, N
26    SB=SB+P(I)*H(I, J)
      SB=SB/SA
      DO 27 I=1, N
27    H (I, J)=H(I, J)-SB*U (I)
25    CONTINUE
      IF (L. EQ. (-3)) GO TO 31
      GO TO 32
28    IT=L
      RETURN
      END
```

```
      SUBROUTINE FCT (X, F)
      DIMENSION X(3), F(3)
      F(1)=X(1)-5.*X(2)**2+7.*X(3)**2+12.
      F(2)=3.*X(1)*X(2)+X(1)*X(3)-11.*X(1)
      F(3)=2.*X(2)*X(3)+40.*X(1)
      RETURN
      END
```

```
      DIMENSION X(3), F(3), P(3), U(3), Y(3), H(3, 3)
      X(1)=1.5
      X(2)=7.5
      X(3)=-6.0
      CALL SNSE4 (3, 1E-5, 100, X, F, IT, IC, P, U, Y, H)
      WRITE (*, 1) IT, X, F
 1    FORMAT (10X,' IT=', I4//, 16X,'X(1)', 12X,'X(2)',
     +12X,' X(3)'/, 8X, 3E16.8//, 12X,'F(2)', 12X,
     +'F(3)'/, 8X, 3E16.8)
      STOP
      END
```

IT=11

X(1)	X(2)	X(3)
.10000000E+01	.60000000E+01	-.40000010E+01
F(1)	F(2)	F(3)
.53405760E-04	.95367440E-06	-.76293950E-05

19. 自适应Runge-Kutta方法

19.1 功能

本程序用自适应(adaptive)Runge-Kutta（龙格-库塔）方法（即自动变步长 Runge-Kutta 方法)求解一阶常微分方程的初值问题

$$\begin{cases} y' = f(t, y) \\ y(t_0) = y_0 \end{cases} \quad ①$$

本程序所用的自适应方法是改进的 Euler（欧拉）方法，即二阶 Runge-Kutta 方法，为了根据局部截断误差自动选取步长，又用到三阶 Runge-Kutta. 局部截断误差的计算估计式是

$$y_{n+1} - Y_{n+1} \approx u_{n+1} - Y_{n+1} \quad ②$$

其中 u_{n+1} 是第 $n+1$ 时间步三阶 Runge-Kutta 的计算值，

$$u_{n+1} = u_n + h(2K_1 + 3K_2 + 4K_3)/9 \quad ③$$

Y_{n+1} 是二阶 Runge-Kutta 的计算值

$$Y_{n+1} = Y_n + hK_2 \quad ④$$

这里

$$K_1 = f(t_n, u_n)$$

$$K_2 = f(t_n + \frac{h}{2}, u_n + \frac{h}{2}K_1)$$

$$K_3 = f(t_n + \frac{3}{4}h, u_n + \frac{3}{4}hK_2)$$

从③式减去④式有

$$EST = \frac{u_{n+1} - Y_{n+1}}{h} = (2K_1 + 4K_3 - 6K_2)/9$$

自动变步长的过程是：当 $EST \leqslant \varepsilon$ 时，下一步仍用 h 计算，当 $EST > \varepsilon$ 时，将 h 缩小一半，下一步用 $\frac{h}{2}$ 计算，ε 是预先给定的局部误差限。

19.2 使用说明

1° 子程序调用语句

CALL RK2ADP (F, TO, T, YO, TOL, HMIN, H, Y, IFLG)

2° 虚元说明

输入参数：

TO 实变量，初始点

T 实变量，计算终止点

YO 实变量，初始函数值

TOL 实变量，局部误差限

HMIN 实变量，允许的最小步长

H 实变量，起始步长

F 外部函数，计算函数 F（T，Y）

输出参数：

Y 实变量，终点的近似函数值

IFLG 整变量，子程序返回方式信息。=1表示正常返回；=-1表示自适应过程失败

19.3 例题

求一阶常微分方程初值问题

$$\begin{cases} \dfrac{dy}{dt} = e^{-t} - y \\ y(0) = 1 \end{cases}$$

的解。

取初始步长 H=0.5，最小允许步长 HMIN=0.05，局部误差限TOL=5×10^{-4}，计算到 $t=1$ 为止。

19.4 程序和计算结果

```
      SUBROUTINE RK2ADP (F, TO, T, YO, TOL,
     +HMIN, H, Y, IFLG)
      REAL K1, K2, K3
      DOUBLE PRECISION YY, DH,  DK
C   INTIALIZATION
      IFLG=0
      TH=TO
      TOL1=.125*TOL
      Y=YO
      YY=YO
    1 TLEFT=T-TH
      IF (H. LE. TLEFT) GOTO 2
C   MAKE CERTAIN THAT FINAL POINT IS T
      H=TLEFT
      IFLG=1
C   MAIN COMPUTATION OF MOD-EULER APPROXIMATION
    2 IF (ABS(H). LE. 1. E-8) GOTO 5
      K1=F(TH, Y)
      K2=F(TH+.5*H, Y+.5*H*K1)
      K3=F(TH+.75*H, Y+.75*H*K2)
      ESTERR=(2*K1+4.*K3-6.*K2)/9.
      EST=ABS(ESTERR)
C   TEST FOR ACCEPTABLE VALUE
      IF (EST. GT. TOL) GOTO 3
C   ACCEPT YY, TAKE NEXT STEP STARTING AT TH+H
      DH=H
      DK=DH*K2
      YY=YY+DK
      Y=YY
      TH=TH+H
      WRITE (*, 99) Y, TH, ESTERR
   99 FORMAT (5X, 3(E12.7, 4X))
C   IFLG=1 SIGNALS THAT T HAS BEEN REACHED
    5 IF(IFLG. EQ. 1) RETURN
C   INCREASE MESHSIZE IF APPROPRIATE
      IF(EST. LT. TOL1) H=2.*H
      GOTO 1
C   STEP REJECTED, RESTART WITH H/2 IF NOT TOO SMALL
    3 H=.5*H
      IFLG=0
      IF (H. GT. HMIN) GOTO 1
C   ADAPTIVE PROCEDURE FAILED
```

```
      IFLG=-1
      RETURN
      END

      REAL FUNCTION F(T, Y)
      F=EXP(-T)-Y
      RETURN
      END

      EXTERNAL F
      DATA TO, YO, H, HMIN, TOL, T/0., 1., 5.,
     +.005, 5.E-4, 1./
      WRITE (*, 66)
   66 FORMAT (10X,' VALUE', 11X,' POINT',
     +10X,' ERREST')
      CALL RK2ADP (F, TO, T, YO, TOL, HMIN,
     +H, Y, IFLG)
      IF (IFLG. GE. 0) STOP
      WRITE (*, 77)
   77 FORMAT (5X,' PREMATURE TERMINATION:H
     +TOO SMALL')
      WRITE (*, 88) H, Y
   88 FORMAT (5X,' FINAL H=', F7.5, 2X,' FINAL
     +Y=', E12.5)
      STOP
      END
```

VALUE	POINT	ERREST
.9995156E+00	.3125000E−01	.2016624E−03
.9981144E+00	.6250000E−01	.1905494E−03
.9958536E+00	.9375000E−01	.1799266E−03
.9927871E+00	.1250000E+00	.1697546E−03
.9889669E+00	.1562500E+00	.1599921E−03
.9844421E+00	.1875000E+00	.1507865E−03
.9792596E+00	.2187500E+00	.1418723E−03
.9734643E+00	.2500000E+00	.1334614E−03
.9670959E+00	.2812500E+00	.1253684E−03
.9602037E+00	.3125000E+00	.1176463E−03
.9528174E+00	.3437500E+00	.1103083E−03
.9449769E+00	.3750000E+00	.1033809E−03
.9367170E+00	.4062500E+00	.9665225E−04
.9280710E+00	.4375000E+00	.9029442E−04
.9190704E+00	.4687500E+00	.8421475E−04

.9087452E+00	.5000000E+00	.7849269E−04
.9001239E+00	.5312500E+00	.7296933E−04
.8902337E+00	.5625000E+00	.6772412E−04
.8801000E+00	.5937500E+00	.6277031E−04
.8697473E+00	.6250000E+00	.5801518E−04
.8484660E+00	.6875000E+00	.2102057E−03
.8265719E+00	.7500000E+00	.1770126E−03
.8042177E+00	.8125000E+00	.1470778E−03
.7815389E+00	.8750000E+00	.1201365E−03
.7586573E+00	.9375000E+00	.9587077E−04
.7356804E+00	.1000000E+01	.7414818E−04

20. 定步长Hamming方法

20.1 功能

本程序采用 Hamming（哈明）方法求解一阶常微分方程组

$$\begin{cases} \dfrac{dy_i}{dy_1} = f_i(y_1, y_2, \cdots, y_n) \\ y_i(y_{10}) = y_{i0} \end{cases} \qquad (i = 1, 2, \cdots, n)$$

的初值问题，其中 $y_1 = t$ 是自变量，$y_{10} = t_0$，$f_1 = 1$.

先用四阶 Runge-Kutta（龙格-库塔）方法从初值 y_{i0}（$i=1, 2, \cdots, n$）开始，步长为 Δt，计算三步，得到 y_{i1}，y_{i2}，y_{i3} 和 y'_{i1}，y'_{i2}，y'_{i3}（$i=1, 2, \cdots n$），为 Hamming 方法提供初值，再用 Hamming 方法继续进行积分.

20.2 使用说明

1° 子程序调用语句

CALL RKHM (N, H, Y, F, M, P, R, A, FCT)

2° 虚元说明

输入参数：

N　　整变量，方程个数

H　　实变量，积分步长

FCT　计算微分方程组右函数的哑过程名. 其形式如下：

SUBROUTINE FCT (N, Y, F)

这里 N 是整变量．方程个数；实数组 Y(N)存放计算右函数用的自变量值；实数组 F(N) 存放右函数值

输入兼输出参数：

M　整变量，作为标志．每当用Runge-Kutta 方法起步时，M为零，其后本子程序自行修改M，使转入 Hamming方法时M为 4．若需再次用 Runge-Kutta方法起步，M 必须为零

Y　N 个元素的一维实数组．开始存放积分初值，最后存放积分结果

输出参数：

F　N 个元素的一维实数组，调用后存放对应于 Y 的同步右函数值

工作单元：

P，R　都是 N 个元素的一维实数组

A　8・N 个元素的二维实数组

3° 附注

在后一次调用本子程序时，需用到前一次调用时存于工作单元 A 中的结果．

自变量看作数组 Y 的第一个分量，其右函数的第一个分量 $f_1=1$，作为第一个方程．因此若原来求解 n 个方程的方程组，就变成了求解 $n+1$ 个方程的方程组．

20.3 例题

求解方程组

$$\begin{cases} y_1' = y_2 \\ y_2' = -y_1 \\ y_1(0)=0 \ , \ y_2(0)=1 \end{cases}$$

初值问题的积分区间 $0 \leqslant t \leqslant 1$，步长 $h=0.1$．

求解时，把方程组改写为：

$$
\begin{cases}
\dfrac{dt}{dt}=1 \\[2ex]
\dfrac{dy_1}{dt}=y_2 \\[2ex]
\dfrac{dy_2}{dt}=-y_1
\end{cases}
$$

取 N=3，Y (1)表示自变量 t，Y (2) 表示 y_1，Y(3)表示 y_2。

20.4 程序和计算结果

```
      SUBROUTINE RKHM (N, H, Y, F, M, P, R, A, FCT)
      DIMENSION U(5), Y(N), F(N), P(N), R(N), A(8, N)
      IF (M. EQ. 4) GOTO 60
      M=M+1
      IF(M. EQ. 1) GOTO 40
      U(1)=0.5*H
      U(2)=U(1)
      U(3)=H
      U(4)=H
      U(5)=U(1)
      DO 10 K=1, N
      A (8, K)=0.
10    P(K)=Y(K)
      DO 30 J=1, 4
      DO 20 K=1, N
      R(K)=P(K)+U(J)*F(K)
20    Y(K)=Y(K)+U(J+1)*F(K)/3
      IF (J. EQ. 4) GOTO 40
30    CALL FCT (N, R, F)
40    CALL FCT (N, Y, F)
      DO 50 K=1, N
      A(1, K)=A(2, K)
      A(2, K)=A(3, K)
      A(3, K)=A(4, K)
      A(4, K)=Y(K)
      A(5, K)=A(6, K)
      A(6, K)=A(7, K)
50    A(7, K)=F(K)
      RETURN
```

```
60    DO 70 K=1, N
      P(K)=A(1, K)+8./3.*H*(A(7, K)-0.5*A(6, K)+A(5, K))
70    R(K)=P(K)-112./121.*A(8, K)
      CALL FCT (N, R, F)
      DO 80 K=1, N
      C=(9.*A(4, K)-A(2, K)+3.*H*
     +(F(K)+2.*A(7, K)-A(6, K)))/8.
      A(8, K)=P(K)-C
80    Y(K)=C+9./121.*A(8, K)
      GOTO 40
      END


      DIMENSION Y(3), F(3), P(3), R(3), A(8, 3)
      EXTERNAL FCT
      Y(1)=0.
      Y(2)=0.
      Y(3)=1.
      M=0
      WRITE (*, 20)
20    FORMAT (10X, 1HT, 14X, 2HY1, 14X, 2HY2)
30    CALL RKHM(3, 0.1, Y, F, M, P, R, A, FCT)
      WRITE (*, 40) Y
40    FORMAT (F15.6, 2F15.5)
      IF (Y(1). LT. 0.95) GOTO 30
      STOP
      END


      SUBROUTINE FCT (N, Y, F)
      DIMENSION Y(N), F(N)
      F(1)=1
      F(2)=Y(3)
      F(3)= -Y(2)
      RETURN
      END
```

T	Y1	Y2
.000000	.00000	1.00000
.100000	.09983	.99500
.200000	.19867	.98007
.300000	.29552	.95534

.400000	.38942	.92106
.500000	.47943	.87758
.600000	.56464	.82534
.700000	.64422	.76484
.800000	.71736	.69671
.900000	.78333	.62161
1.000000	.84147	.54030

参考资料

〔1〕 李庆扬、王能超、易大义编，《数值分析》，华中工学院出版社，第三版，1987年。

〔2〕 清华大学、北京大学《计算方法》编写组编，《计算方法》（上、下册），1974年，1980年。

〔3〕 冯康等编，《数值计算方法》，国防工业出版社，1978年。

〔4〕 王仁宏著，《数值有理逼近》，上海科技出版社，1980年。

〔5〕 李岳生、黄友谦，《数值逼近》，人民教育出版社，1978年。

〔6〕 徐利治、王仁宏、周蕴时，《函数逼近的理论与方法》，上海科技出版社，1983年。

〔7〕 李庆扬等，《非线性方程数组解法》，科学出版社，1987年。

〔8〕 蔡大用，《数值代数》，清华大学出版社，1987年。

〔9〕 陆金甫、关治，《偏微分方程数值解法》，清华大学出版社，1987年。

〔10〕 李荣华、冯果忱，《偏微分方程数值解法》，人民教育出版社，1980年。

〔11〕 姜礼尚、庞之恒，《有限元方法及其理论基础》，人民教育出版社，1979年。

〔12〕 南京大学数学系，《常微分方程数值解法》，科学出版社，1979年。

〔13〕 刘德贵等编，《FORTRAN 算法汇编》（第一、二分

册》，国防工业出版社，1980年，1983年。

〔14〕 上海机械学院、安徽省计算中心，《FORTRAN 实用程序库》，上海科技文献出版社，1984年。

〔15〕 北京大学、吉林大学、南京大学编，《计算方法》，高等教育出版社，1961年。

〔16〕 Gourlay, A.R., and G. A.watson (1980), 《矩阵特征值问题的计算方法》，唐焕文等译，上海科学技术出版社。

〔17〕 Hageman, L. A., and D. M. Young (1984), 《实用迭代法》，蔡大用、施妙根译，清华大学出版社。

〔18〕 Stewart, G. W. (1980), 《矩阵计算引论》，王国荣等译，上海科学技术出版社。

〔19〕 Szidarovszky, F., and S. Yakowitz (1982), 《数值分析的原理及过程》，施明光、潘仲雄译，上海科学技术文献出版社。

〔20〕 Tewarson, R. P. (1981), 《稀疏矩阵》，朱季纳译，科学出版社。

〔21〕 Henrici, P. (1985), 《常微分方程的离散变量方法》，包雪松等译，科学出版社。

〔22〕 Gear, C. W. (1978),《常微分方程初值问题的数值方法》，费景高等译，科学出版社。

〔23〕 Forsythe, G.E., and C. B. Moler (1979), 《线性代数方程组的计算机解法》，徐树荣译，科学出版社。

〔24〕 Adey, R. A., and C. A. Brebbia (1983), Basic Computational Techniques for Engineers, Pentech Press.

〔25〕 Atkinson, K. E.(1978),An Introduction to Nu-

merical Analysis, Wiley, New York.

[26] Burden, R. L., J. D. Faires and A. C. Reynolds (1981), Numerical Analysis, 2nd ed, Weber, Boston, Massachusetts.

[27] Demidovich, B. P., and I. A. Maron (1981), Computational Mathematics, Mir Publishers, Mosscow.

[28] Churchhouse, R. F. (1981), Handbook of Applicable Mathematics, Wiley, New York.

[29] Isaacson, E., and H. B. Keller (1966), Analysis of Numerical Methods, Wiley, New York.

[30] Keller, H. B. (1968), Numerical Methods for Two-Point Boundary Value Problems, Ginn-Blaisdell.

[31] King, J. T. (1984), Introduction to Numerical Computation, McGraw-Hill Book Company.

[32] Kopchenova, N. V., and I. A. Maron (1981), Computational Mathematics, Mir Publisbers, Moscow.

[33] Lambert, J. D. (1976), Computational Methods in Ordinary Differential Equations, Wiley, New York.

[34] Nakamura, S. (1977), Computational Methods in Engineering and Science with Applications to Fluid Dynamics and Nuclear Systems, Wiley, New York.

[35] Noye, J. (1984), Computational Technique for Differential Equations, Elsevier, New York.

[36] Ortega, J. M., and W. C. Rheinboldt (1970),

Iterative Solution of Nonlinear Equations in Several variable, Academic Press, New York.
[37] Varga, R. S.(1962), Matrix Iterative Analysis, Prentice-Hall, Inc, Englewood Cliffs, New Jersey.
[38] Wilkinson, J.(1963), Rounding Error in Algebraic Processes, Notes on Applied Science No. 32, Her Majesty's Stationary Office, London; Prentice-Hall, Inc, Englewood Cliffs, New Jersey.
[39] Wilkinson, J. (1965), The Algebraic Eigenvalue Problems, Clarendon Press, Oxford.

中文-外文索引

一划

二划

三划

四划

五划

六划

七划

八划

九划

十划

十一划

十二划

十三划

十四划

十五划

十六划

外文-中文索引

A

B

C

D

E

F

G

H

I

J

K

L

N

O

Q

R

T

U

V

W

外国人名表

Adams, J.C. 亚当斯
Aitken, A.C. 埃特金
Bairstow, L. 贝尔斯多夫
Bashforth, F. 巴诗福斯
Bernoulli 伯努利
Bessel, F.W. 贝塞尔
Brent, P.R. 布兰特
Brown, G.W. 布朗
Broyden, C.G. 布罗依登
Bui, T.D. 布意
Cauchy, A.L. 柯西
Cholesky 乔莱斯基
Cohen, C.J. 柯亨
Concus, P. 康卡斯
Cotes, R. 柯特斯
Courant, R. 库朗
Cramer, G. 克莱姆
Crank, J. 克兰克
Crout, P.D. 克劳特
D'Alembert, J.L.R. 达朗贝尔
Davidon, W. 达维东
Day, M.M. 戴依
Dirichlet, P.G.L. 狄利希莱特
Doolittle 杜利特尔

Henrici, P. 亨里奇
Hermite, C. 埃尔米特
Hesse, H. 赫赛
Hessenberg, K. 赫申伯格
Heun, K. 休恩
Hilbert, D. 希尔伯特
Hollingsworth 荷令斯沃思
Hölder, O. 荷尔德
Horner, W.G. 霍纳
Householder, A.S. 豪斯荷尔德
Huta, A. 休塔
Jacobi, C.G.J. 雅可比
Jordan, C. 约当
Krawczyk, R. 克拉夫楚克
Kroneker, L. 克隆内克
Kuhn, H. 库恩
Kutta, W. 库塔
Lagrange, J.L. 拉格朗日
Laguerre, E.N. 拉盖尔
Lanczos, C. 兰错斯
Laplace, P.S.M. 拉普拉斯
Lax, P.D. 拉克斯
Legendre, A.M. 勒让德
Lewy, H. 勒维
Liebman, K.O.H. 李普曼
Lipschitz, R.O.S. 李普希茨
Lobatto 罗巴托
Maclaurin, C. 马克劳林
Maehly 梅利

Merrill, O. 默里尔
Milne, W.E. 米尔尼
Moore, R. 莫尔
Morrison, D. 莫里森
Moulton, F.R. 莫尔顿
Muller, D.E. 密勒
Newton, I. 牛顿
Nickel, K. 尼克尔
Nicolson, P. 尼科尔松
Nyström, E.J. 尼斯特龙
Padé, H. 巴德
Patterson, L.N. 帕特松
Peacemen, D.W. 俾斯曼
Poisson, S.D. 泊松/普阿松
Pouzet 布泽
Powell, M.B. 保韦尔
Rachford, H.H. 瑞奇福尔德
Radau, R. 拉道
Ralston, A. 赖尔斯顿
Rayleigh, J.W.S. 瑞利
Rémès, E.J. 里姆斯
Riccati, J.F. 黎卡提
Richardson, L.F. 李查逊
Riemann, G.F.B. 黎曼
Ritz, W. 里兹
Robinson 罗宾森
Roberts, K.V. 罗伯茨
Romberg, W. 龙贝格
Rolle, M. 罗尔

Runge, C. 龙格
Ruth 卢斯
Scarf, H. 斯卡夫
Schwartz, H.A. 施瓦兹
Seidel, P.L. 赛德尔
Shanno, D.F. 香诺
Sherman, A. 谢门
Simpson, T. 辛甫生
Sperner, E. 斯珀内
Steffensen, J.F. 斯梯芬森
Stirling, J. 斯特林
Sturm, J.C.F. 斯特姆
Taylor, B. 泰勒
Thiele, T.N. 蒂埃勒
Vorterra, V. 沃尔泰拉
Von Neumann, J. 冯·诺依曼
Weierstrass, K. 维尔斯特拉斯
Weiss, N.O. 维斯
Wendroff, B. 温德洛夫
Widlund, O. 怀德伦德
Woodburg 伍德伯格
Бернштейн, С.Н. 伯恩斯坦
Галёркин, Б.Г. 伽辽金
Канторович, Л.В. 康托洛维奇
Марков, А.А. 马尔可夫
Чебышев, П.Л. 切比雪夫
Шаманский, В. 沙曼斯基
Яненко 扬年科